UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

Desarrollo de aplicaciones web con el API de Google Cloud

MEMORIA PRESENTADA POR:

Ana Olcina Valero

GRADO DE *INGENIERÍA INFORMÁTICA*

Convocatoria de defensa: *Julio 2017*

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
CAMPUS D'ALCOI

Desarrollo de aplicaciones web con el API de Google Cloud

Ana Olcina Valero

Resumen
Las aplicaciones web en la nube (cloud) proporcionan la posibilidad de optimizar los recursos, romper barreras geográficas y aumentar la productividad de las empresas. Además Google, que proporciona una gran cantidad de servicios de usuario, también ofrece servicios cloud de empresa con el claro objetivo de aumentar su productividad. Este aumento en la productividad se debe, básicamente a dos motivos. Por un lado, tenemos la reducción de la complejidad en los desarrollos (temas como la seguridad o escalabilidad son soportados directamente por Google). Por el otro lado, está el abaratamiento de los costes, al reducir la inversión en infraestructuras informáticas (no es necesario aprovisionar ni mantener ningún servidor) utilizando las propias de Google.
En base a lo anteriormente expuesto, el objetivo de este trabajo de fin de grado es doble: (i) realizar un estado del arte exhaustivo al respecto de los servicios que proporciona Google Cloud y (ii) llevar a cabo la implementación de una aplicación web utilizando los servicios de Google que resulten más adecuados.

Dirección
Pau Micó

CONTENIDO

1. Introducción

1.1 Antecedentes

Tras completar todas las asignaturas del grado de Ingeniería Informática, de la Universidad Politécnica de Valencia en el Campus de Alcoy, realizo el siguiente trabajo de final de grado, cuya **propuesta** es el análisis de los servicios cloud de empresa que proporcionan los principales proveedores que existen actualmente, centrándome principalmente en Google, así como la implementación de una aplicación web utilizado dichos servicios.

El **motivo principal** por el que he decidido realizar mi trabajo fin de grado relacionado con el desarrollo de aplicaciones web con el API de Google Cloud, viene determinado por el deseo de aprender y profundizar en el conocimiento del desarrollo de aplicaciones en una plataforma moderna, en constante actualización y que considero que tiene un gran futuro. Por otra parte, considero que este trabajo me permite aplicar los conceptos fundamentales adquiridos durante el grado a una solución real.

1.2 Objetivos

- ➢ Realizar un **estado del arte** exhaustivo al respecto de los servicios que proporciona Google Cloud.
- ➢ Llevar a cabo la **implementación de una aplicación web** utilizando los servicios de Google que resulten más adecuados, con la finalidad principal de aprender y explicar el procedimiento a seguir para realizar una aplicación web utilizando Google Cloud Platform.

1.3 Requerimientos

El cliente, una red de clínicas veterinarias y hospitales veterinarios, solicita una aplicación web para poder ofrecer a sus clientes un nuevo servicio que proporcione la posibilidad de que sus clientes registren la información de las visitas veterinarias que realizan con sus mascotas.

La seguridad de los datos es uno de los requisitos más importantes a tener en cuenta, por ello todos los usuarios de la aplicación deberán autenticarse en el sistema para poder utilizar el servicio. Teniendo en cuenta esto, los usuarios se registrarán y actualizarán su perfil con sus datos personales y darán de alta a sus mascotas. Posteriormente, podrán registrar los datos más significativos de cada visita, así como se establecerá la fecha de la próxima visita la cual la podrán añadir a su propio calendario de Google.

Otros requisitos a tener en cuenta serán: una alta fiabilidad del sistema con una baja tasa de fallos, tiempos de respuesta rápidos y una alta usabilidad.

Inicialmente, se implantará la aplicación para unos centros veterinarios en concreto, pero se necesita que la aplicación sea escalable para poderla implantar en toda la red veterinaria, sin que esto suponga una gran inversión económica.

La aplicación se llamará "Centro Veterinario".

2. Estado del arte

Según [1], la nube o computación en la nube (*Cloud Computing*) es la plataforma tecnológica por excelencia en la década actual y, posiblemente, del futuro de la computación.
La nube (*Cloud*) es el conjunto de servidores de información desplegados en centros de datos, a lo largo de todo el mundo, donde se almacenan millones de aplicaciones Web y enormes cantidades de datos a disposición de miles de organizaciones y empresas, y cientos de miles de usuarios que se descargan y ejecutan directamente los programas y aplicaciones de software almacenados en dichos servidores tales como Google, Amazon, IBM o Microsoft.
La nube está propiciando una nueva revolución industrial, soportada en los Centros de Datos (*Data Centers*), y de aplicaciones Web (*Web Apps*). Esta nueva revolución industrial ya está produciendo un gran cambio social, tecnológico y económico.

Hoy en día, las aplicaciones web en la nube proporcionan la posibilidad de optimizar los recursos, romper barreras geográficas y aumentar la productividad de las empresas. Este aumento en la productividad se debe, básicamente a dos motivos:

- reducción de la complejidad en los desarrollos, ya que temas como la seguridad o escalabilidad son soportados directamente por los proveedores de estos servicios
- abaratamiento de los costes, reduciendo la inversión en infraestructuras informáticas ya que al utilizar las propias de los proveedores, no es necesario aprovisionar ni mantener ningún servidor, además se paga por el uso de los servicios cloud contratados

El modelo de desarrollo, dentro del cloud computing, que permite implementar aplicaciones sin preocuparnos de la infraestructura informática es el de Plataforma como Servicio (*Platform as a Service - PaaS*), el cual proporciona una plataforma con las APIs de desarrollo para que se implementen las aplicaciones, que se ejecutarán en los sistemas proporcionados por el proveedor.
Actualmente, existe una gran cantidad de proveedores de este tipo de servicios pero, en este estado del arte, me voy a centrar en las compañías que lideran este mercado, las cuales son:

- Google: con su plataforma de servicios cloud *"Google Cloud Platform".*
- Amazon: con su plataforma de servicios cloud *"Amazon Web Services".*
- Microsoft: con su plataforma de servicios cloud *"Microsoft Azure".*

2.1 Google

Google proporciona una plataforma con todos los servicios para la computación en la nube llamada ***Google Cloud Platform (GCP)*** [2], la cual ofrece toda la infraestructura necesaria para poder crear y escalar aplicaciones dependiendo de las necesidades de cada momento.

2.1.1 Google Cloud Platform

En esta plataforma encontramos una gran cantidad de servicios [3] agrupados en diversas áreas como pueden ser:

- **Recursos informáticos:** nos proporciona la posibilidad de crear nuestras propias máquinas virtuales a medida a través de *Compute Engine* o bien utilizar la plataforma de desarrollo de aplicaciones totalmente administrada denominada *App Engine*. [4]

- **Almacenamiento y bases de datos:** nos permite almacenar objetos a través del servicio *Cloud Storage* y bases de datos, escalables y de alto rendimiento, tanto bases de datos relacionales con *Cloud SQL* como bases de datos NoSQL con *Cloud Bigtable o Cloud Datastore*. [5]

- **Redes:** nos ofrece productos de red definidos mediante software de última generación en la red de fibra privada de Google, como por ejemplo *Cloud Virtual Network, Cloud Load Balancing* o *Cloud DNS*, entre otros, y que nos proporcionan la funcionalidad de gestión de la red para los recursos de Google Cloud Platform, balanceo de carga escalable y de alto rendimiento y servicio DNS de baja latencia de la red mundial de Google, respectivamente [6]

- **Big Data:** podremos disponer de almacenamiento de datos totalmente administrados con *BigQuery*, lotes y procesamientos de secuencia de datos con *Cloud Dataflow*, exploración de datos con *Cloud Datalab* (el cual se integra con *Cloud BigQuery* y *Cloud Machine Learning*), gestionar grandes conjuntos de datos Hadoop y Spark a través de *Cloud Dataproc* y mensajería en tiempo real confiable a través de *Cloud Pub/Sub*. [7]

- **Aprendizaje automático:** suministra servicios de Machine Learning rápidos y escalables, con modelos ya creados o con la posibilidad de crear nuevos modelos a partir de los datos. Por ejemplo, para poder realizar análisis de texto con *API Natural Language*, o de imágenes con *API Vision* o incluso reconocimiento de voz con *API Speech* [8]

- **Herramientas de administración:** podremos llevar a cabo una supervisión en tiempo real, así como diversos diagnósticos mediante *Stackdriver*, una consola de administración que nos proporciona diferentes servicios para poder saber en todo momento el estado de nuestras aplicaciones. También existen otro tipo de servicios como *Cloud Deployment Manager* que permite simplificar el despliegue de aplicaciones o *Cloud Console* que nos ayudará a implementar, escalar y diagnosticar problemas a través de una interfaz basada en web o incluso a través de una aplicación móvil [9]

- **Herramientas para desarrolladores:** podremos encontrar diversas herramientas y librerías que ayudarán a desarrollar e implementar nuestras aplicaciones de una forma productiva, a través de la interfaz de línea de comandos y otras herramientas para desarrolladores, como puede ser el conjunto de librerías y herramientas dentro de *Cloud SDK*, los repositorios Git privados a través de *Cloud Source Repositories*, las herramientas cloud para Android Studio o diversos complementos para Eclipse. [10]
- **Identidad y seguridad:** podremos controlar el acceso y la visibilidad de los recursos en una plataforma protegida por el modelo de seguridad de Google, gestionar claves de cifrado, analizar las aplicaciones para detectar vulnerabilidades comunes y todo ello a través de servicios como: *Cloud IAM, Cloud Key Management Service, Cloud Resource Manager, Cloud Security Scanner.* [11]

Figura 1. *Oferta de servicios de Google Cloud*

Como se puede observar, Google dispone de una gran cantidad de servicios para cloud pero, en este estado del arte, me voy a centrar en *Google App Engine* ya que proporciona todos los servicios necesarios para posteriormente desarrollar la aplicación web que presento en este trabajo sobre una gran infraestructura como es la de Google.

2.1.2 Google App Engine

Google App Engine [12] es una plataforma como servicio que proporciona la posibilidad de crear nuestras propias aplicaciones y backends móviles y escalarlas automáticamente dependiendo de la cantidad de tráfico recibido, de esta forma únicamente se paga por los recursos que se utilicen en cada momento sin necesidad de aprovisionar ni mantener ningún servidor.

Además, cuenta con APIs y servicios integrados como bases de datos relacionales y/o NoSQL, servicio de almacenamiento distribuido en memoria caché para mejorar el rendimiento de las aplicaciones, una API de autenticación de usuarios, balanceo de carga, comprobaciones de estado y registros de la aplicación, así como análisis y detección automática de problemas de seguridad más habituales.

Se pueden utilizar las herramientas de desarrollo más conocidas, como Eclipse, IntelliJ, Maven, Git, Jenkins y PyCharm e implementar las aplicaciones utilizando lenguajes populares como Java, Python, PHP o Go.

App Engine **ofrece dos entornos de ejecución**: un entorno estándar y un entorno flexible (en versión beta). Aunque, actualmente, en Europa no está soportado el entorno flexible voy a comentar brevemente las características de ambos entornos.

El **entorno estándar** [13-14] usa contenedores predefinidos de Google para envolver el código y lo ejecuta en la infraestructura de Google. Los contenedores predefinidos están disponibles para los siguientes lenguajes de programación: Java 7, Python 2.7, PHP 5 y Go.

El **entorno flexible** [14-15] usa contenedores Dockers para envolver el código y lo ejecuta en la infraestructura de Google Compute Engine, por lo que se podrá personalizar la infraestructura sobre la cual se ejecutarán las aplicaciones y, por tanto, se podrá ajustar el rendimiento de las mismas. Incluye soporte nativo para Java 8, Python 2.7 y Python 3.5, Node.js, Ruby, PHP 5 y PHP 7 y Go, pero además se pueden personalizar esos lenguajes de programación o incluso proporcionar otros propios.

Entre ambos entornos existen otras **diferencias** [16] como pueden ser el tiempo de inicio de instancia que es mayor en el entorno flexible, al igual que sucede con el tiempo máximo de espera de solicitud. El entorno flexible, además proporciona procesos en segundo plano, depuración SSH y soporta la instalación de binarios de terceros. Estas y otras diferencias hacen que su precio también sea diferente, el cual en el entorno estándar está basado en las horas de instancias mientras que el del entorno flexible está basado en el uso de CPU virtual, memoria y discos persistentes.

Pero hay que tener en cuenta que la **elección del entorno**, en mi caso, está **determinado por la ubicación**, ya que tanto el entorno estándar como el flexible están disponibles en América del Norte y en Asia-Pacífico, pero el entorno flexible no está soportado en Europa.

Antes de pasar a detallar **las características** de App Engine, hay que tener en cuenta que **se clasifican según su estado y disponibilidad** en: Alfa, Beta y Disponibilidad General [13]. Estas últimas están cubiertas por la cobertura del acuerdo del nivel de servicio *(Service Level Agreement - SLA)* [17] que, a grandes rasgos, nos asegura que se proporcionará un porcentaje de disponibilidad del servicio del 99.95% y si Google no cumple con esto, el cliente podrá recibir unos determinados reembolsos los siguientes meses. Y por otro lado, todas las características pueden no estar disponibles en todos los lenguajes de programación e incluso que la funcionalidad sea distinta.

A continuación, de detallan las **características más significativas** del entorno estándar de App Engine: [13]

- Usa el estándar **Java Servlet 2.5** para aplicaciones web.
- Su *almacén de datos,* **Cloud Datastore**, es un almacén de datos NoSQL el cual nos permite albergar objetos de datos, denominados entidades y proporciona alta disponibilidad de lecturas y escrituras y puede ejecutar varias operaciones en transacciones atómicas.
- **Google Cloud SQL** es una base de datos MySQL en la nube con prácticamente todas las capacidades y funcionalidades de MySQL y alguna adicional.
- La **API Blobstore** permite a las aplicaciones servir objetos de datos, llamados blobs, y son útiles para almacenar archivos de mayor tamaño que el permitido en el almacén de datos, por ejemplo, imágenes y vídeos.
- La **API de búsqueda** proporciona un modelo de indexación de documentos que contienen los datos estructurados. Los documentos e índices se guardan en un almacén persistente, separado, y optimizado para operaciones de búsqueda. Una sóla búsqueda puede devolver hasta 10.000 documentos coincidentes.
- Proporciona un **servicio de memoria caché** de los datos (no persistente), llamado **Memcaché**, que mejorará el rendimiento de las aplicaciones. Podrá ser memoria caché compartida con el resto de aplicaciones (gratuita) o dedicada que se asignará exclusivamente a una aplicación (se pagará por GB/hora).
- La **API de Registros** proporciona un acceso a los registros de la aplicación, aunque también podremos acceder a ellos a través de la consola de Google Cloud Platform.
- Las aplicaciones **podrán enviar mensajes de correo electrónico** en nombre de los administradores de la aplicación y en nombre de los usuarios con cuentas de Google, **así como recibir** correos electrónicos en diferentes direcciones.
- A través de la **extracción de URL** las aplicaciones podrán comunicarse con otras o acceder a otros recursos en la web a través de URLs. Este servicio utiliza la infraestructura de red de Google para mejorar la eficiencia y escalabilidad.
- Con la **API de cola de tareas**, las aplicaciones pueden realizar tareas fuera de una petición del usuario, útil, por ejemplo, si una aplicación necesita ejecutar alguna tarea en segundo plano.
- El **servicio Cron** de App Engine permite configurar diferentes tareas programadas para operar en momentos definidos o en intervalos regulares.
- La API de Imágenes proporciona la capacidad de describir y manipular los datos de las imágenes.
- Pueden **autenticar a los usuarios** de las cuentas de Google o de las cuentas en sus propios dominios de Google App. Una aplicación podrá detectar si el usuario actual ha iniciado sesión y redirigirlo a la página correspondiente, por lo que será útil a hora de implementar las áreas de administración de la aplicación.
- **Es regional** [18], es decir, la infraestructura sobre la que se ejecutan las aplicaciones está localizadas en una región específica y están gestionadas por Google para estar disponibles en todas las zonas dentro de esas regiones. Actualmente, App Engine está disponible en las siguientes regiones: Estados Unidos - Central, Estados Unidos - Este, Europa - Oeste y Asia - Noreste.

- **Proporciona 1 GB de almacenamiento y tráfico de datos de forma gratuita**, aunque se puede aumentar dependiendo de las necesidades.
- **Cada aplicación se ejecuta en una instancia de una clase diferente**, la cual determinará sus recursos informáticos y precios. La siguiente tabla muestra las diferentes instancias que existen actualmente:

Instancia de Clase	**Límite de Memoria**	**Límite de CPU**
B1	128 MB	600 MHz
B2	256 MB	1.2 GHz
B4	512 MB	2.4 GHz
B4_1G	1024 MB	2.4 GHz
B8	1024 MB	4.8 GHz
F1	128 MB	600 MHz
F2	256 MB	1.2 GHz
F4	512 MB	2.4 GHz
F4_1G	1024 MB	2.4 GHz

2.2 Amazon

El siguiente proveedor elegido es Amazon, que proporciona una plataforma con todos los servicios para la computación en la nube llamada ***Amazon Web Services (AWS)*** [19], la cual ofrece también toda la infraestructura necesaria para poder implementar y ejecutar aplicaciones web, e incluso escalarlas dependiendo de las necesidades de cada momento.

2.2.1 Amazon Web Services

De forma similar a Google, en esta plataforma encontramos una gran cantidad de servicios [20] agrupados en diversas áreas:

- **Computación:** suministra servidores virtuales en la nube a través de *Amazon EC2*, la posibilidad de ejecutar y gestionar contenedores Docker con *Amazon EC2 Container Service* e incluso almacenar y recuperar imágenes Docker con *Amazon EC2 Container Registry*, así como ejecutar y administrar aplicaciones web mediante su plataforma *AWS Elastic Beanstalk.* [21]
- **Almacenamiento:** proporciona almacenamiento escalable en la nube con *Amazon S3*, almacenamiento de archivos administrados para EC2 con *Amazon Elastic File System* o almacenamiento de archivos en la nube a bajo costo con *Amazon Glacier.* [22]
- **Bases de datos:** dispone de servicios de bases de datos relacionales administrado para MySQL, PostgreSQL, Oracle, SQL Server y MariaDB en *Amazon RDS* y, también, para bases de datos NoSQL administradas en *Amazon DynamoDB*. [23]
- **Migración:** se pueden realizar migraciones de servidores con *AWS Server Migration Service* o bases de datos con tiempo de inactividad mínimo a través de *AWS Database Migration Service*, e incluso transferencias de datos a escala de petabytes o de exabytes, con *AWS Snowball* o *AWS Snowmobile*, respectivamente. [20]
- **Redes:** permite aprovisionar una sección de la nube de Amazon Web Services (AWS) aislada de forma lógica utilizando *Amazon VPC*, distribuir automáticamente el tráfico entrante de las aplicaciones entre varias instancias de Amazon EC2 en la nube y conseguir niveles altos de tolerancia a errores mediante *Elastic Load Balancing* y tener un servicio web DNS escalable y de alta disponibilidad como es *Amazon Route 53.* [24]
- **Herramientas para desarrolladores:** conjunto de servicios diseñados para ofrecer una mejor forma de trabajo a los equipos de desarrollo, como pueden ser *AWS CodeCommit* que permite almacenar código en repositorios privados, *AWS CodeDeploy* para automatizar la implementación de código, *AWS CodeBuild* para compilar y realizar pruebas de código, o la *interfaz de línea de comandos de AWS* que sirve para administrar los servicios de AWS. [25]
- **Herramientas de administración:** conjunto de servicios que ayudan a los administradores a aprovisionar, configurar, administrar y monitorizar los recursos de su infraestructura, como por ejemplo, *Amazon EC2 Systems Manage*r para configurar y administrar instancias EC2 y servidores, *Amazon CloudWatch* para monitorizar recursos y aplicaciones o *AWS CloudTrail* para realizar seguimientos de la actividad de los usuarios y el uso de las API. [26]

- **Seguridad, identidad y conformidad:** proporciona diversos servicios para poder trabajar con la mayor seguridad en la infraestructura a través de *AWS Identity & Access Management* para administrar el acceso de usuarios y claves de cifrado, *Amazon Inspector* para analizar la seguridad de las aplicaciones, *AWS Certificate Manager* para aprovisionar, administrar e implementar certificados SSL/TLS o *AWS WAF* para filtrar tráfico web malintencionado. [27]
- **Análisis:** varios servicios ofrecen la capacidad de extraer conocimiento e información práctica de los datos, como pueden ser *Amazon Athena* o *Amazon EMR*, para consultar datos en S3 con SQL, procesar grandes volúmenes de datos a través de un marco Hadoop administrado, respectivamente. [28]
- **Inteligencia artificial:** existen servicios que permiten crear interfaces de conversación para las aplicaciones con voz y texto con *Amazon Lex*, añadir el análisis de imágenes a las aplicaciones con *Amazon Rekonigtion*, convertir texto en habla realista que se puede almacenar y reproducir con *Amazon Polly* y crear aplicaciones utilizando tecnología de aprendizaje automático con *Amazon Machine Learning.* [29]
- **Servicios móviles:** relacionados con esta área encontramos servicios orientados al desarrollo de aplicaciones móviles como es *AWS Mobile Hub* que permite crear, probar y monitorizar aplicaciones, *AWS Device Farm* para probar aplicaciones Android, Fire OS e iOS en dispositivos reales en la nube y también dispone de una kit de desarrollo de software móvil llamado *SDK para móviles AWS*. [30]
- **Mensajería:** en este apartado encontramos *Amazon SES* que es un servicio de envío y recepción de correo electrónico o *Amazon SNS* que es un servicio de notificaciones automatizado, entre otros. [20]
- **Internet de las cosas:** servicios como *AWS Greengrass* y *AWS IoT* podemos recopilar y enviar datos a la nube para su posterior análisis e incluso permitir a los dispositivos conectados interactuar con aplicaciones y otros dispositivos. [31]
- **Desarrollo de juegos:** proporciona un motor de juegos gratuito multiplataforma en 3D con código fuente completo e integrado con AWS y Twitch denominado *Amazon Lumberyard*. [32]

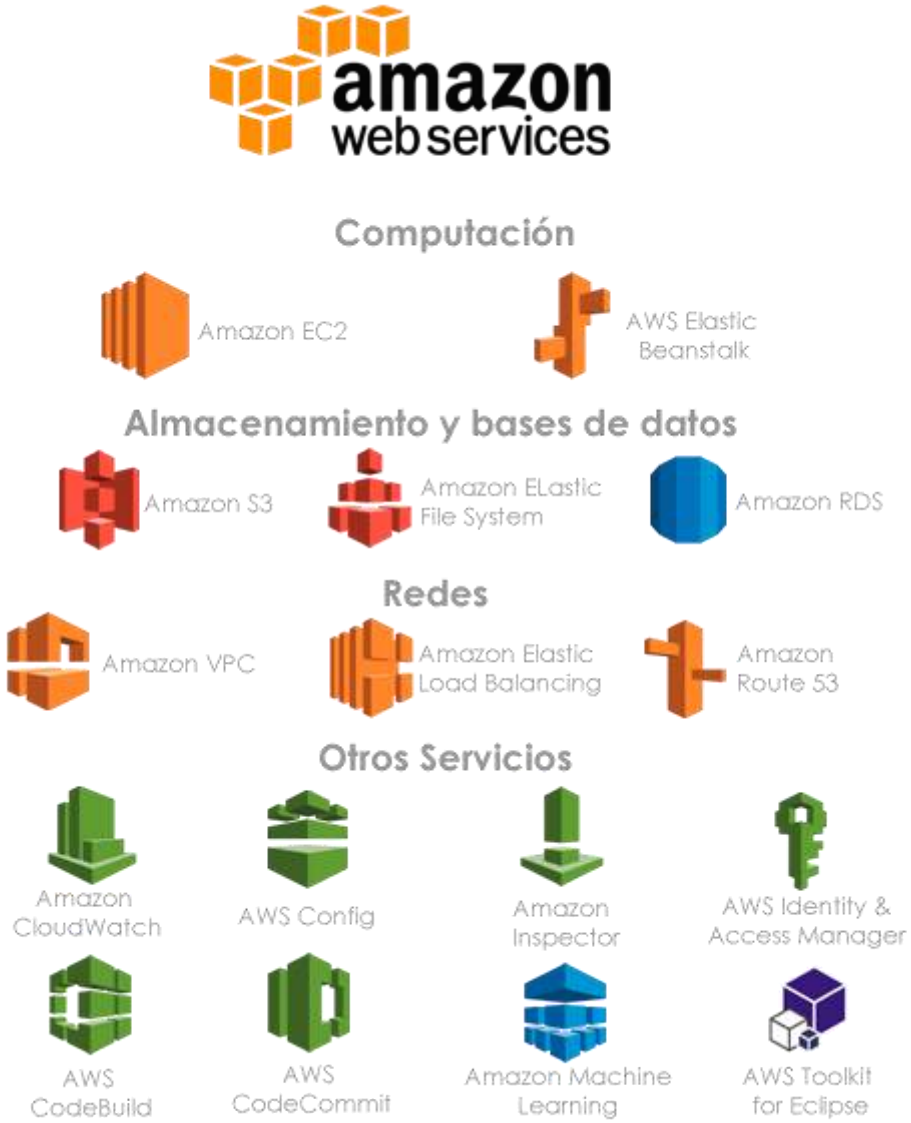

Figura 2. *Oferta de servicios de Amazon Web Services*

Al igual que ocurría con la plataforma de Google, Amazon ofrece una gran cantidad de servicios para cloud pero únicamente me voy a centrar en describir, en el siguiente punto, las propiedades de *AWS Elastic Beanstalk* que se sería el servicio comparable con *App Engine* de Google y es el que proporciona todos los servicios necesarios para poder desarrollar una aplicación web sin preocuparnos del aprovisionamiento de la infraestructura necesaria.

2.2.2 AWS Elastic Beanstalk

AWS Elastic Beanstalk [33] es una plataforma como servicio que proporciona la posibilidad de implementar y gestionar nuestros propios servicios y aplicaciones web desarrolladas con Java, .Net, PHP, Node.js, Python, Ruby, Go y Docker en servidores como Apache, Nginx, Passenger e IIS. Además AWS proporciona conjuntos de herramientas especializadas para la nube integradas en los entornos de desarrollo Eclipse y Visual Studio.

Únicamente cargando el código (por ejemplo, un archivo .war de Java), Elastic Beanstalk se encargará de llevar a cabo una administración automática, desde el aprovisionamiento de la capacidad, el equilibrio de carga y el escalado automático hasta la monitorización del estado de la aplicación, pero tendremos acceso a los recursos subyacentes cuando lo necesitemos.

En este caso sólo se paga por los recursos de AWS que se necesiten para almacenar y ejecutar las aplicaciones.

Las **particularidades más significativas** de Elastic Beanstalk son: [34 - 35]

- Para cada lenguaje de programación habrá diferentes **tipos de contenedores**, dichos contenedores definirán la pila de infraestructura y software para ser utilizado en un entorno determinado. La pila de software que se ejecuta en las instancias de Amazon EC2 depende del tipo de contenedor (por ejemplo, para aplicaciones de 32 bits o de 64 bits).
- Cuando iniciamos un **entorno Elastic Beanstalk** se elige:
    - El tipo de entorno de ejecución: determina si Elastic Beanstalk provisiona recursos para dar soporte a una aplicación web que se encarga de HTTP(S), en este caso se denomina un entorno *"Web Server"* o una aplicación web que se encarga de las tareas de procesamiento en segundo plano denominado entorno *"Worker"*.
    - La plataforma: proporciona plataformas preconfiguradas para lenguajes de programación (Java, PHP, Python, Ruby, Go), contenedores web (Tomcat, Passenger) y contenedores Dockers, con múltiples configuraciones para cada uno. También es compatible con plataformas personalizadas.
- Elastic Beanstalk **provisiona los recursos necesarios** para ejecutar nuestras aplicaciones incluyendo una o más instancias Amazon EC2.
- A través de *AWS Management Console*, las APIs o interfaces de línea de comandos, por ejemplo, *AWS CLI* o *EB*, podremos **obtener información** sobre la aplicación y **gestionarla**.
- Se pueden crear **plantillas de recursos** mediante *AWS CloudFormation* y utilizarlas para el lanzamiento de nuevos recursos de AWS sin tener que personalizarlos cada vez.
- Suministra varias opciones disponibles de **bases de datos y almacenamiento persistente**, como *Amazon RDS, Amazon DynamoDB, Microsoft SQL Server, Oracle, IBM DB2* o *Informix* o un servicio de **almacenamiento en memoria caché**.
- Ofrece **múltiples servicios adicionales**, por ejemplo, un servicio de mensajería para poder ser notificados en las aplicaciones llamado *Amazon Simple Service Notification (Amazon SNS)*, un servicio para monitorizar el estado de la aplicación llamado *CloudWatch* o un servicio para gestionar las métricas y los umbrales para determinar cuándo añadir o eliminar instancias denominado *Auto Scaling.*

2.3 Microsoft

El tercer proveedor elegido es Microsoft, que como en los casos anteriores, proporciona una plataforma con todos los servicios necesarios para la computación en la nube llamada ***Microsoft Azure.*** [36]

2.3.1 Microsoft Azure

De forma similar a Google y a Amazon, en esta plataforma encontramos una gran cantidad de servicios [37] agrupados en diversas áreas:

- **Computación:** proporciona diversos servicios para aprovisionar máquinas virtuales de Windows y Linux *(Virtual Machine)*, administrar y escalar a múltiples máquinas virtuales (*Conjuntos de escalado de máquinas virtuales*), crear aplicaciones web y móviles para cualquier plataforma y dispositivo *(App Service)*, usar herramientas basadas en Docker para implementar y administrar contenedores *(Container Service)* e incluso guardar imágenes de dichas implementaciones de contenedores *(Azure Container Registry)* así como ejecutar trabajos de procesos en lotes y paralelos de gran escala *(Batch)* o crear aplicaciones y API en la nube de alta disponibilidad y escalabilidad *(Cloud Services)*.
- **Redes:** en cuanto a los servicios relacionados con las redes, encontramos la posibilidad de aprovisionar redes privadas y, si es necesario, conectarse a centros de datos locales *(Virtual Network)*, conseguir un alto rendimiento de red y una alta disponibilidad para las aplicaciones *(Load Balancer)*, establecer conectividad segura entre entornos locales *(VPN Gateway)* o administrar el tráfico entrante para un alto rendimiento y disponibilidad *(Traffic Manager)*.
- **Almacenamiento:** suministra almacenamiento en la nube, de alta disponibilidad y escalable a gran escala *(Storage)* y dentro de este servicio encontramos almacenamiento de objetos para datos no estructurados *(Blob Storage)*, escalado de aplicaciones según el tráfico (Queue Storage), recursos compartidos de archivos que utilicen el protocolo estándar SMB 3.0 *(File Storage)*. Así como, un servicio de repositorio a gran escala para cargas de trabajo de análisis de macrodatos *(Data Lake Store)* o copias de seguridad de los servidores en la nube *(Backup)*.
- **Web y móvil:** proporciona diversos servicios para crear e implementar aplicaciones web críticas *(Web Apps)* o crear y hospedar el servidor back-end para cualquier aplicación móvil *(Mobile Apps)*, entre otros.
- **Bases de datos:** ofrece servicios administrados para bases de datos relacionales *(SQL Database)*, para documentos NoSQL *(DocumentDB)* o para conjuntos de datos semiestructurados a través de un almacén de clave-valor *(Table Storage)*.
- **Inteligencia + Análisis:** dispone de un servicio para poder aprovisionar clústeres de Hadoop, Spark, R Server, HBase y Storm en la nube *(HDInsight)*, un servicio de análisis distribuido *(Data Lake Analytics)*, herramientas de análisis predictivo *(Machine Learning)* así como agregar funcionalidades de API inteligentes para habilitar interacciones contextuales *(Cognitive Services)*.
- **Internet de las cosas:** ofrece la posibilidad de conectar, supervisar y controlar miles de activos de IoT *(Centro de IoT de Azure)* así como procesar flujos de datos en tiempo real procedentes de dispositivos IoT *(Stream Analytics).*

- **Seguridad + Identidad:** se puede evitar y detectar amenazas y responder a ellas (*Azure Security Center*) así como administrar identidades y accesos de consumidores *(Azure Active Directory B2C)* e incluso proteger el acceso a los datos y aplicaciones con un nivel adicional de autenticación *(Multi-Factor Authentication)*.
- **Supervisión + Administración:** dispone de varios servicios para llevar a cabo tareas de supervisión y administración, por ejemplo, se puede compilar, administrar y supervisar todos los productos de Azure a través de una consola unificada *(Portal de Microsoft Azure)*, administrar los recursos de la aplicación *(Azure Resource Manager)*, detectar, evaluar y diagnosticar problemas en los servicios y aplicaciones web *(Application Insights)* o ejecutar trabajos programados recurrentes *(Scheduler)*.
- **Herramientas para desarrolladores:** suministra herramientas para los equipos de desarrolladores, para compartir código, supervisar el trabajo y distribuir software *(Visual Studio Team Services)* así como implementar, compilar, diagnosticar y administrar aplicaciones y servicios multiplataforma escalables *(Herramientas y SDK para desarrolladores).*

Figura 3. *Oferta de servicios de Microsoft Azure*

Al igual que en los casos previos, de todos los servicios que ofrece Microsoft Azure únicamente me voy a centrar en especificar las características de *App Service* ya que proporciona todo lo necesario para crear y trabajar con nuestras propias aplicaciones web.

2.3.2 App Service

App Service [38] es la oferta de plataforma como servicio de Microsoft Azure, la cual nos permite crear aplicaciones web y móviles para cualquier dispositivo, ya que incluye ambas funcionalidades.

Azure ejecuta las aplicaciones en máquinas virtuales totalmente administradas, con la elección de máquinas virtuales con recursos compartidos o máquinas virtuales dedicadas (esto estará determinado por el plan de precios que elijamos) en la cuales podremos crear distintos tipos de aplicación utilizando varios lenguajes y plataformas ya que es compatible con ASP.NET, Node.js, Java, PHP y Python. Además, existen diversas herramientas dedicadas en Visual Studio que permiten optimizar las tareas de creación, implementación y depuración de las aplicaciones.

A continuación, se indican algunas de las **características y funcionalidades principales** de App Service:

- Ofrece **varios tipos de aplicación**, cada uno dedicado a hospedar una carga de trabajo específica:
  - Aplicaciones web: recursos de procesos que Azure proporciona para hospedar sitios y aplicaciones web.
  - Mobile Apps: recursos de procesos que Azure proporciona para hospedar back-ends de aplicaciones móviles.
  - API Apps: recursos de procesos que Azure proporciona para hospedar API de RESTful.
  - Logic Apps: recursos de procesos que Azure proporciona para automatizar procesos empresariales e integrar sistemas y datos.
- Los **planes de App Service** representan la colección de recursos físicos que se utilizan para hospedar las aplicaciones. Estos planes definen lo siguiente:
  - Región: existen varias regiones ubicadas en América, Europa y Asia-Pacífico.
  - Recuento de escala: número de instancias: una, dos, tres…
  - Tamaño de la instancia: pequeña, mediana, grande.
  - Planes de precio del servicio (SKU): gratis, compartido, básico, estándar y premium. Se podrá escalar de un SKU gratis y compartido a uno básico, estándar o premium para poder acceder a más recursos y características. Además, si establecemos el básico o superior, se podrá controlar el tamaño y recuento de escala de las máquinas virtuales.

  El SKU y la escala es lo que determinará el costo, no el número de aplicaciones en un entorno.
- Además de aplicaciones en los lenguajes de programación indicados, también se puede ejecutar **Windows PowerShell y otros scripts o ejecutables** en máquinas virtuales de App Service.
- Nos permite la **integración e implementación continua** a través de Visual Studio Team Services, GitHub o BitBucket.

- Se podrá llevar a cabo la **administración de las aplicaciones** a través de Azure PowerShell o de la interfaz de la línea de comandos (CLI) multiplataforma.
- Suministra la **posibilidad de escalar** de forma automática o manual e incluso hospedar las aplicaciones en cualquier parte de la infraestructura del centro de datos global de Microsoft y el Acuerdo de Nivel de Servicio de App Servicio promete **alta disponibilidad** (igual que en el caso de Google App Engine, está establecido en como mínimo un 99.95% como porcentaje de disponibilidad del servicio).
- Ofrece la **posibilidad de conectar** a sistemas empresariales, plataformas SaaS y a datos locales a través de diversos conectores, conexiones híbridas y/o Azure Visual Networks.

2.4 Otras opciones

Para finalizar con este estado del arte, aunque me he centrado en las soluciones de cloud computing propietarias de diversas empresas comerciales, también hay que comentar que existen alternativas de código abierto (*Open Source*) [39], como pueden ser: *OpenShift* [40], *Cloud Foundry* [41], *Tsuru* [42] o *Cloudify* [43], entre otras.

De todas ellas, me voy a centrar en **OpenShift**, la plataforma como servicio de código abierto que ofrece **Red Hat**, la cual automatiza el hosting, configuración, implementación y administración de las aplicaciones en la nube. Además, permite gestionar el tráfico y la demanda para conseguir una alta escalabilidad y alta disponibilidad.

2.4.1 OpenShift

A continuación, se detallan algunas de sus **características más importantes**: [39]

- OpenShift se basa en **tecnologías** como **Red Hat Enterprise Linux®** y **Red Hat JBoss® Middleware.**
- La plataforma OpenShift está disponible en dos **modelos de consumo diferentes**:
  - **PaaS pública (*OpenShift Online*)** [44]: es el servicio público de alojamiento de la plataforma PaaS que ofrece una solución de desarrollo, compilación, implementación y alojamiento de aplicaciones en nube y que permite que las aplicaciones escalen de forma automática en un entorno de nube flexible.
    Ofrece compatibilidad con diversos lenguajes de programación como: .NET, Python, Java, PHP, Node.js, Ruby y Perl.

    Suministra almacenes de datos relacionales y NoSQL, como pueden ser: MySQL, PostgreSQL, MongoDB o MariaDB.

    Además de los lenguajes de programación y servicios incorporados, los desarrolladores pueden añadir otros componentes de lenguajes, bases de datos o middleware que necesiten a través de *OpenShift Cartridge API*.

    Incluye el sistema de gestión de control de versiones de código Git y nos proporciona repositorios de código preconfigurados para diversos lenguajes, bases de datos o frameworks.

Admite la creación de aplicaciones en diferentes regiones, a través de varias zonas geográficas consiguiendo menores latencias mediante el despliegue de las aplicaciones en los servidores más cercanos de los usuarios previstos.

Suministra un servicio de instantáneas para realizar copias de seguridad y restauración.

Dispone de contenedores seguros de código (*Gears*) que asigna CPU, memoria, disco y ancho de banda para las aplicaciones y con la posibilidad de configurar las aplicaciones para escalar automáticamente en respuesta del tráfico web.

Posee una arquitectura con un plano de control sin estado (Brokers), una infraestructura de mensajería y una infraestructura de alojamiento de aplicaciones (nodos) que además se podrá configurar con redundancia para eliminar los impactos de fallos de hardware.

También incluye una interfaz de consola web para desarrolladores para que estos puedan crear, modificar y gestionar sus aplicaciones pero, además, tiene una integración en Eclipse, JBoss Developer Studio y Titanium Studio.

- **PaaS privada (*OpenShift Container Platform*)** [45]: que utiliza la misma plataforma PaaS de código abierto que ejecuta el servicio alojado OpenShift Online y lo almacena para los clientes que deseen una implementación en nube privada o local. Y que además proporciona las siguientes características adicionales:
  - Añadir lenguajes de programación adicionales.
  - Conectar y ampliar los servicios locales en un centro de datos.
  - El clúster Openshift sólo contendrá a nuestros usuarios, aplicaciones y servicios a través de una plataforma aislada.
  - Disponibilidad regional global.
  - Infraestructura gestionada y con soporte técnico 24x7.

Según Red Hat, la **ventaja** que ofrecen las plataformas de código abierto es que se puede controlar la implementación y de esta forma no se está limitado al trayecto tecnológico y empresarial de un proveedor en concreto, así como la posibilidad de colaborar con otras comunidades y empresas para **fomentar la innovación** en las áreas que consideremos más importantes.

Finalmente, hay que tener en cuenta que **una plataforma esté basada en código abierto no significa que sea gratuita**, ya que existen diferentes planes de precios de Red Hat que incluyen diversos recursos dependiendo de nuestras necesidades como por ejemplo CPU, memoria y/o almacenamiento extra, así como soporte técnico.

3. Anteproyecto

3.1 Estudio de propuestas

Para poder desarrollar la aplicación web, teniendo en cuenta todos los requerimientos del cliente, se propone diseñar un **sistema distribuido con una arquitectura multicapa** basada en el paradigma cliente-servidor.

Para poder llevarlo a cabo, se implementará un **proyecto web dinámico de Java con Maven** utilizando el entorno de desarrollo Eclipse.

En cuanto al EIS (*Enterprise Information System,* backend) se optará por un **modelo de datos NoSQL** ya que proporciona la escalabilidad y el rendimiento necesario para este tipo de proyecto. Principalmente se ha elegido este modelo de datos porque vivimos en una era en la que se trabaja con enormes cantidades de datos y a consecuencia de esto han surgido modelos de almacenamiento de información alternativos a las bases de datos relacionales, por lo que he considerado interesante conocer el funcionamiento y las ventajas de los mismos.

Además, se pretende que el proyecto sea soportado en su totalidad por algunos de los proveedores de cloud descritos anteriormente en el estado del arte.

A continuación, se detallan las características de las implementaciones con los tres principales proveedores de cloud que existen en la actualidad.

3.1.1 Implementación con Google

Para llevar a cabo la solución propuesta, en el caso de Google, la plataforma que albergará la aplicación será **App Engine** con los siguientes **servicios principales**:

- Cloud DataStore para el almacenamiento.
- Las APIs propias de Google (por ejemplo, para autenticar usuarios y añadir eventos al calendario).
- Logging para llevar a cabo el control de registros de la aplicación.
- La cola de tareas para gestionar las tareas de la aplicación.

La **arquitectura** que se utilizará para la aplicación web seguirá el esquema que ofrece Google App Engine [46]:

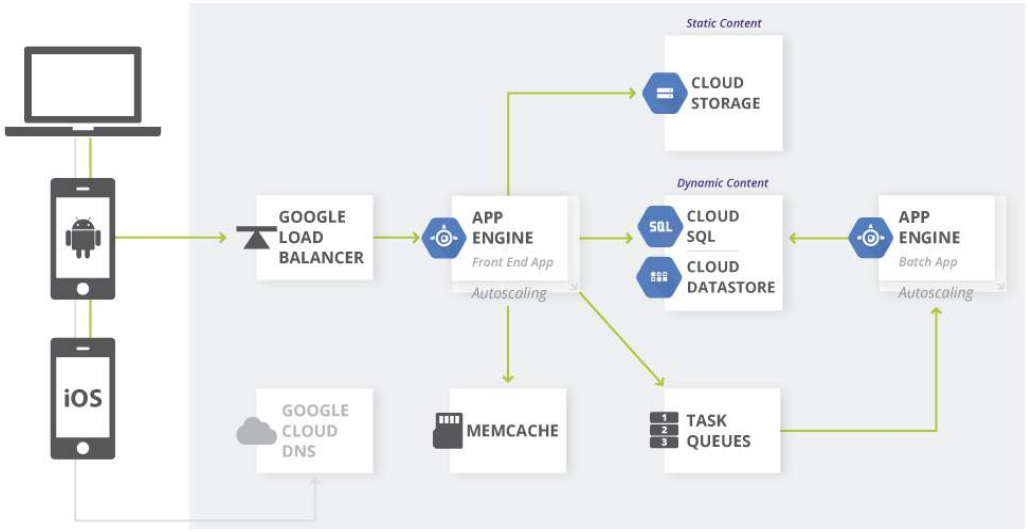

Figura 4. *Arquitectura de una aplicación Web en Google App Engine*

Como ya se ha comentado en el estado del arte, el beneficio de utilizar este tipo de plataformas es que se paga dependiendo de las necesidades y del consumo y, en este caso, encontramos un gran abanico de **precios** [47] **y cuotas** [48] para poder elegir la opción que más se adapte al proyecto. También hay que tener en cuenta que Google nos ofrece un **periodo de prueba gratuito** anteriormente de 2 meses y ahora ampliado a 12 meses [49], con un crédito de 300 dólares estadounidenses, en el cual podremos llevar a cabo la implementación inicial y las correspondientes pruebas y posteriormente se pasaría a contratar los servicios necesarios. Google Cloud Platform nos ayuda a estimar el coste total antes de llevar a cabo nuestra aplicación ya que ofrece una **calculadora de precios** dependiendo de nuestras necesidades [50].

Una gran ventaja que encontramos es la de poder utilizar en nuestro proyecto las **APIs propias de Google** [51]. El ejemplo clásico es el del problema de la autenticación de usuarios. Para ello, la aplicación que desarrollemos con Google se puede beneficiar del hecho de que las cuentas de usuario de Google están muy extendidas (la mayoría de los posibles usuarios poseen ya una cuenta de Google) por lo que no tendrán que realizar ningún tipo de registro especial en la nueva app ya que se utilizará para ello la cuenta de Google que ya posee el usuario. Otro ejemplo de facilidad de uso es el de la sincronización de eventos de la app con el calendario propio del usuario.

Además, otro de los puntos fuertes de Google es que proporciona una gran **documentación para los desarrolladores** a nivel general de los servicios de Google Cloud Platform [52] como en el caso de App Engine [53], así como ofrece completos **cursos gratuitos** en Coursera [54] y Udacity [55], los que he encontrado más interesantes han sido uno para poder tener una visión general de la plataforma de cloud

de Google [56] y otro para aprender diseñar aplicaciones web escalables, entender la envergadura que tendrá un proyecto Java en su plataforma y todo lo necesario para llevarla a cabo [57].

El **entorno de la plataforma Google Cloud Platform** es sencillo e intuitivo que permite seleccionar y/o crear nuevos proyectos desde la parte superior, tener acceso de una forma rápida a todos los servicios disponibles desde su menú lateral, y en la parte central aparecerán las diversas opciones del servicio seleccionado. Además, la ubicación de los elementos de la parte central es personalizable y los podemos distribuir a nuestro gusto y desde la esquina de cada uno de ellos podemos acceder a su propia documentación.

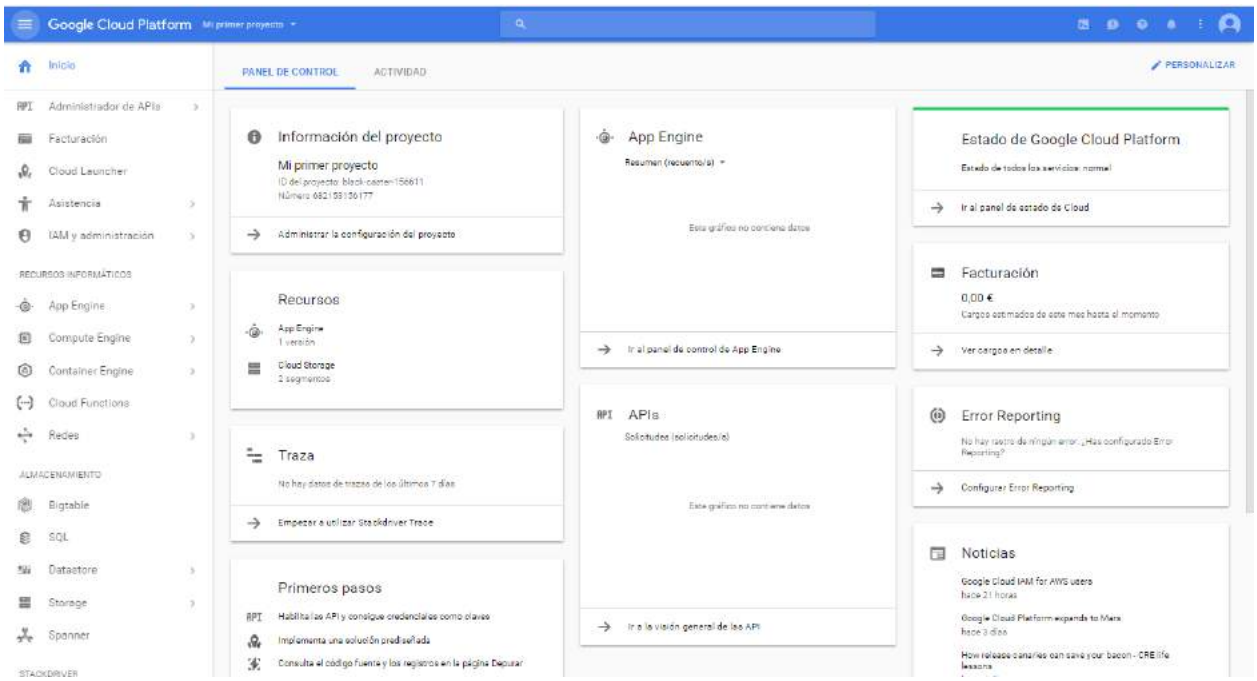

Figura 5. *Captura de pantalla de la plataforma Google Cloud Platform.*

En cuanto al servicio de **App Engine**, también nos muestra una interfaz sencilla y de fácil uso, con acceso inmediato a todos los recursos y servicios disponibles y todo en español.

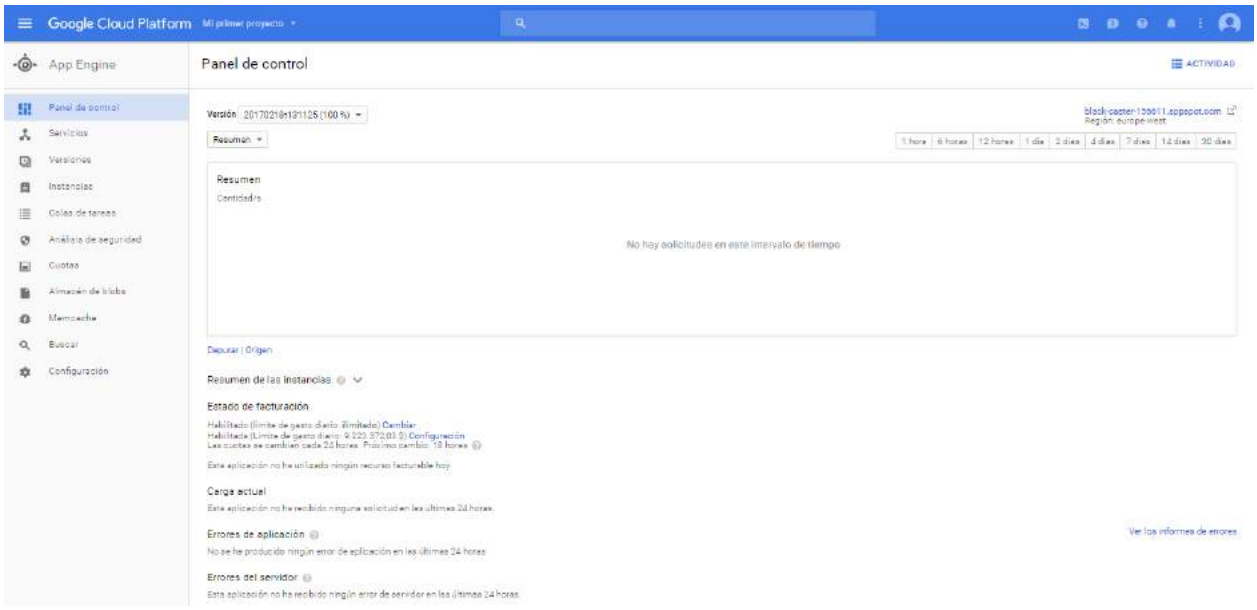

Figura 6. *Captura de pantalla de App Engine.*

Además de toda la documentación y cursos que ofrece Google para su plataforma de Cloud, cuando se **crea la primera aplicación** con App Engine te ofrece la posibilidad de realizar un pequeño **tutorial interactivo** para aprender a crear la primera aplicación con el lenguaje de programación seleccionado previamente.

App Engine nos ofrece un **subdominio** de appspot.com o la posibilidad de utilizar un dominio propio [58]. En el caso del dominio que nos ofrece Google, nuestra aplicación tendrá la siguiente **URL**: http://id-del-proyecto.appspot.com. Por tanto, dependiendo del identificador de nuestro proyecto, la URL de nuestra aplicación web será una más o menos clara y fácil de recordar.

A nivel de desarrollo, también hay que tener en cuenta que podremos utilizar el **plugin** que Google ofrece **para Eclipse** [59], facilitando el trabajo y las pruebas a la hora de implementar y sincronizar el proyecto.

Como inconvenientes, nos encontramos con que únicamente se podrá hacer uso del entorno estándar ya que el flexible no está disponible en Europa y nos limita a la **versión 7 de Java**.

También hay que ser conscientes de que existe la **posibilidad de un fallo en la infraestructura de Google** que nos dejaría sin servicio.

3.1.2 Implementación con Amazon

En el caso de Amazon, para llevar a cabo la solución propuesta, la plataforma que albergará la aplicación será **AWS Elastic Beanstalk** con los siguientes **servicios principales**:

- Instancia EC2 con su grupo de seguridad.
- Amazon S3 bucket para el almacenamiento del código fuente y registros.
- DynamoDB (servicio de base de datos NoSQL).

La siguiente imagen representa una **arquitectura general** de los diferentes componentes y sus interacciones de una aplicación AWS Elastic Beanstalk a través de entornos y servicios AWS [60]:

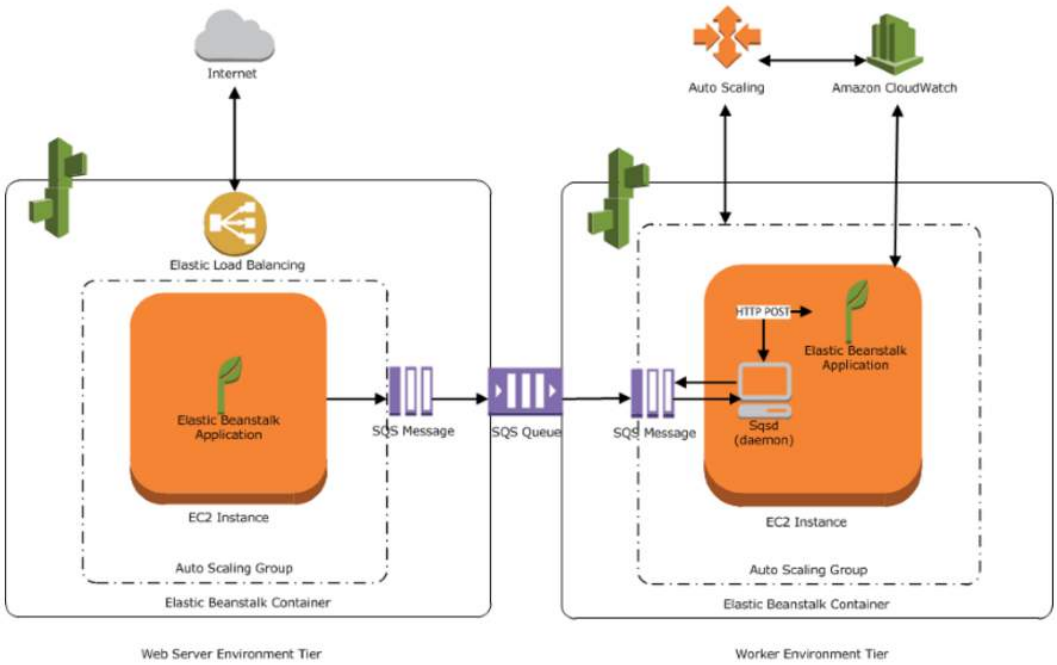

Figura 7. *Diagrama de AWS.*

Una diferencia encontrada con respecto a Google y que supone un inconveniente, es que, aunque existe gran cantidad **documentación para los desarrolladores** en ocasiones no es fácil llegar hasta ciertos temas concretos, y en otras ocasiones la documentación está muy segmentada en diferentes sitios y esto provoca que se pierda mucho tiempo. Un ejemplo concreto, ha sido el poder encontrar la arquitectura de una aplicación web con los servicios que interactúa y una vez encontrado el diagrama representativo es bastante genérico comparado con el que nos ofrece Google o Microsoft. Otro aspecto negativo a tener en cuenta es que, si se crea una aplicación en AWS utilizando uno de los ejemplos disponibles, dichas aplicaciones únicamente están disponibles durante un tiempo determinado, luego ya no estarán disponibles.

El **entorno de la plataforma de Amazon Web Services** es muy diferente al que ofrece Google, aunque ambos cuentan con un estilo sencillo, en el caso de Amazon Web Services debes acceder cada vez desde la parte superior al servicio deseado.

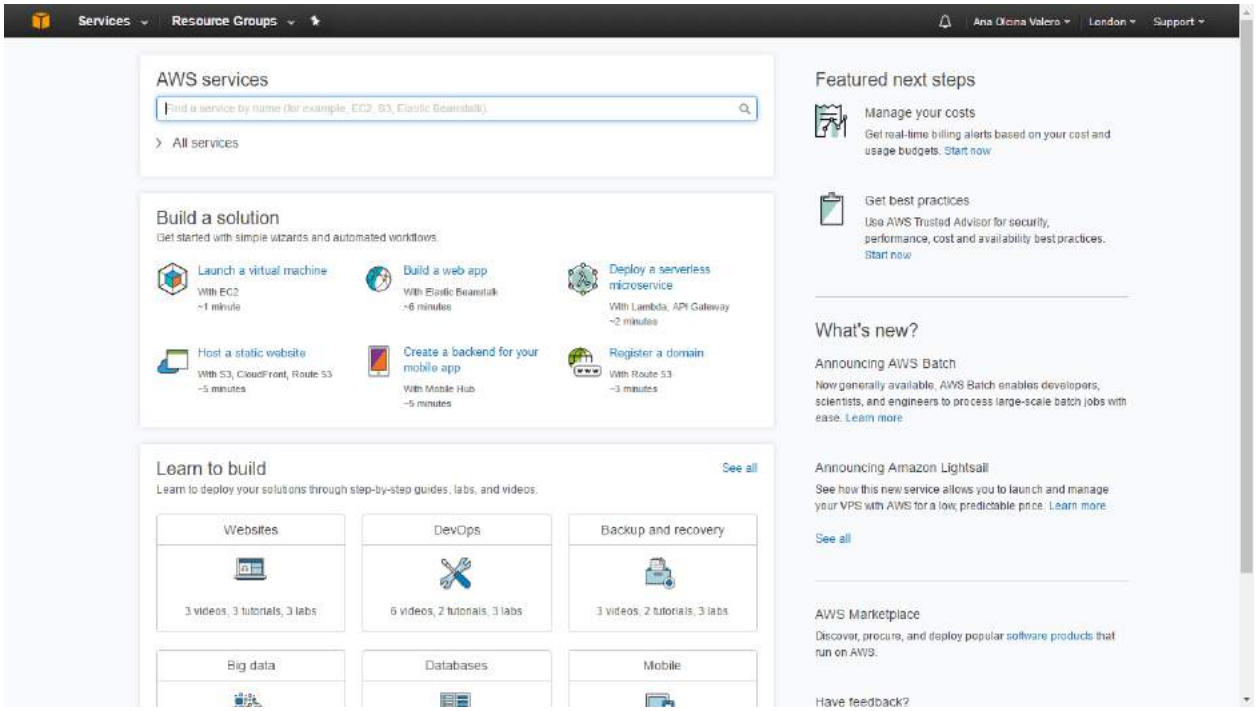

Figura 8. *Captura de pantalla de la plataforma Amazon Web Services*

Una vez accedes a **AWS Elastic Beanstalk** encuentras una interfaz con acceso rápido para poder crear una nueva aplicación, aunque en cuanto al idioma, está todo en inglés.

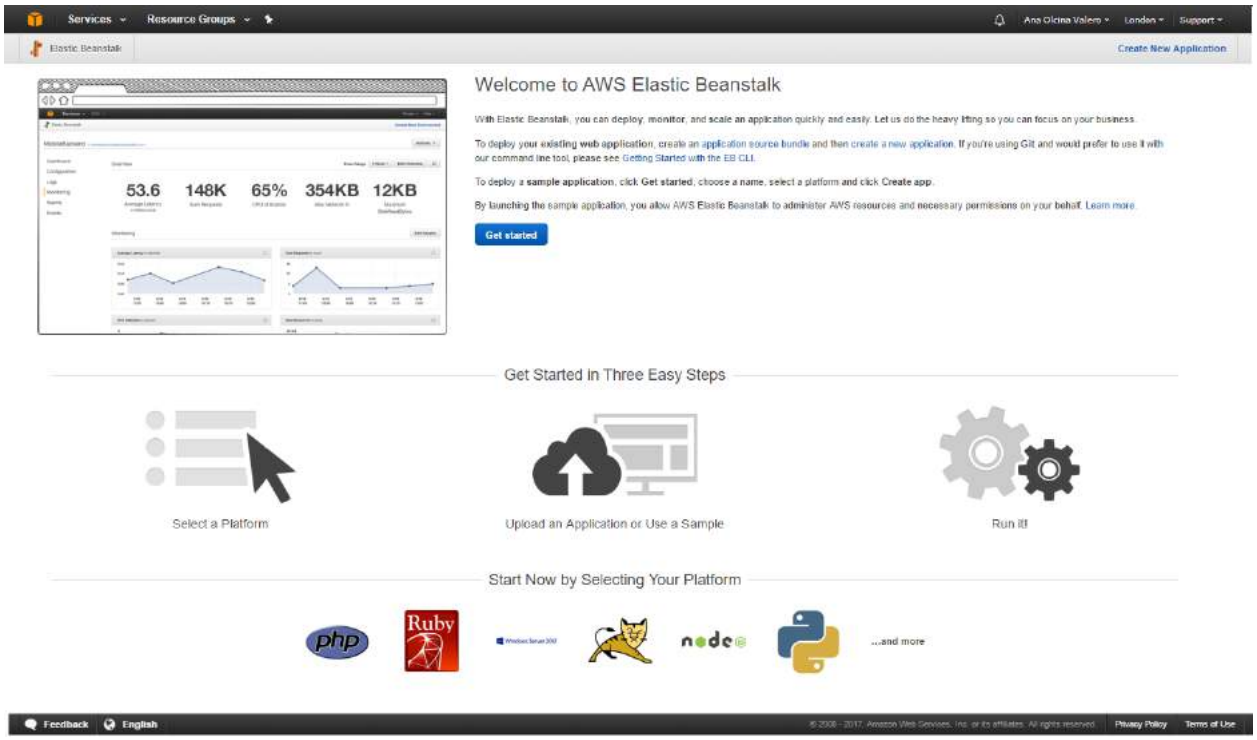

Figura 9. *Captura de pantalla de AWS Elastic Beanstalk*

A la hora de **crear una aplicación**, también es diferente de cómo se crea en App Engine, además de crear la aplicación, también debes crear el entorno de ejecución teniendo que elegir la plataforma y el tipo de aplicación a crear (si va a ser una aplicación de ejemplo, vamos a subir el código o es una actualización).

Mientras se está creando la aplicación nos aparecen unos vínculos para poder acceder a documentación de ayuda, pero en ningún caso es un tutorial interactivo y en cuanto el entorno de la aplicación se crea redirige a otra página y ya no tienes acceso a eso vínculos directamente.

Una vez esté creado el entorno de ejecución, Amazon ofrece un **subdominio** de elasticbeanstalk.com pero también podremos utilizar nuestro propio dominio configurándolo desde Route 53 [61]. En el caso del dominio que nos ofrece Amazon, la aplicación tendrá un identificador aleatorio y esto generará una **URL** poco clara y difícil de recordar del estilo de los siguientes ejemplos:

- http://sample-env-2.gwurps3i37.eu-west-2.elasticbeanstalk.com/
- http://lowcost-env.m7r4qefmnr.eu-west-2.elasticbeanstalk.com/

Un beneficio que encontramos al utilizar Amazon es la posibilidad de crear nuestra aplicación con la **versión 8 de Java**.

A nivel de desarrollo, también hay que tener en cuenta que podremos utilizar el **plugin** que Amazon ofrece **para Eclipse** [62], facilitando el trabajo y las pruebas a la hora de implementar el proyecto.

Como ya se ha comentado previamente, otro de los beneficios de utilizar este tipo de plataformas es que se paga dependiendo de las necesidades y del consumo. En este aspecto, Amazon nos proporciona una **capa de acceso gratuito** [63] durante un año, pasado este tiempo ofrece un **sistema de pago** por uso de servicios individuales [64], en este caso deberíamos pagar por Amazon EC2 [65], Amazon S3 [66] y Amazon DynamoDB [67]. Ya que puede resultar un tanto complejo el definir el coste total a través de los precios de los distintos servicios que necesitemos, Amazon también nos ofrece una **calculadora de precios** [68] que nos permitirá estimar la factura mensual, aunque también ofrece otro tipo de calculadora [69] más avanzada que permite comparar el coste de ejecutar las aplicaciones en un entorno local u hospedado tradicional con el de AWS.

También hay que ser conscientes de que existe la **posibilidad de un fallo en la infraestructura de Amazon** que nos dejaría sin servicio, como ocurrió el día 28 de febrero de este mismo año. En este caso fue un fallo humano, por introducir mal una orden, el que provocó un error en el servicio de almacenamiento S3 del centro de datos de la región del este de Estados Unidos el cual influyó en que diversos servicios de todo el mundo sufrieran una caída o un mal funcionamiento, como fue el caso de Slack, Trello, Sprinklr, Venmo o Down Detector, y se hicieron eco diferentes medios digitales de todo el mundo como CNN [70], Xataka [71] o Expansión [72], entre otros.

Por su parte, Amazon fue informando a través de su cuenta oficial de Twitter:

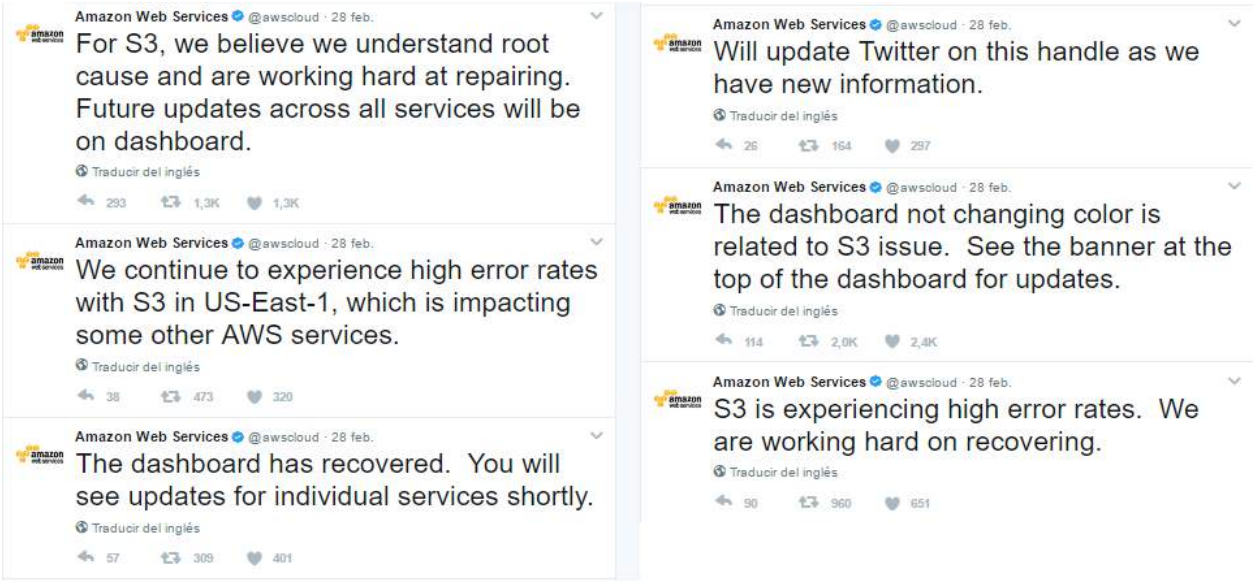

Figura 10. *Capturas de los tuits de Amazon el día 28 de febrero de 2017.*

Y posteriormente publicó un informe detallado sobre la interrupción del servicio en el que aparte de detallar el suceso pidieron disculpas a sus clientes. [73]

3.1.3 Implementación con Microsoft

Finalmente, en el caso de Microsoft Azure, la plataforma que albergará la aplicación será **App Service** con los siguientes **servicios principales**:

- Azure Active Directory para controlar el acceso
- Storage Account para almacenar logs, colas y contenido estático
- Table Storage (Un almacén claves-valores NoSQL para desarrollo rápido que usa conjuntos de datos semiestructurados masivos)

Microsoft Azure propone la siguiente **arquitectura** para proporcionar escalabilidad y rendimiento a una aplicación web [74]:

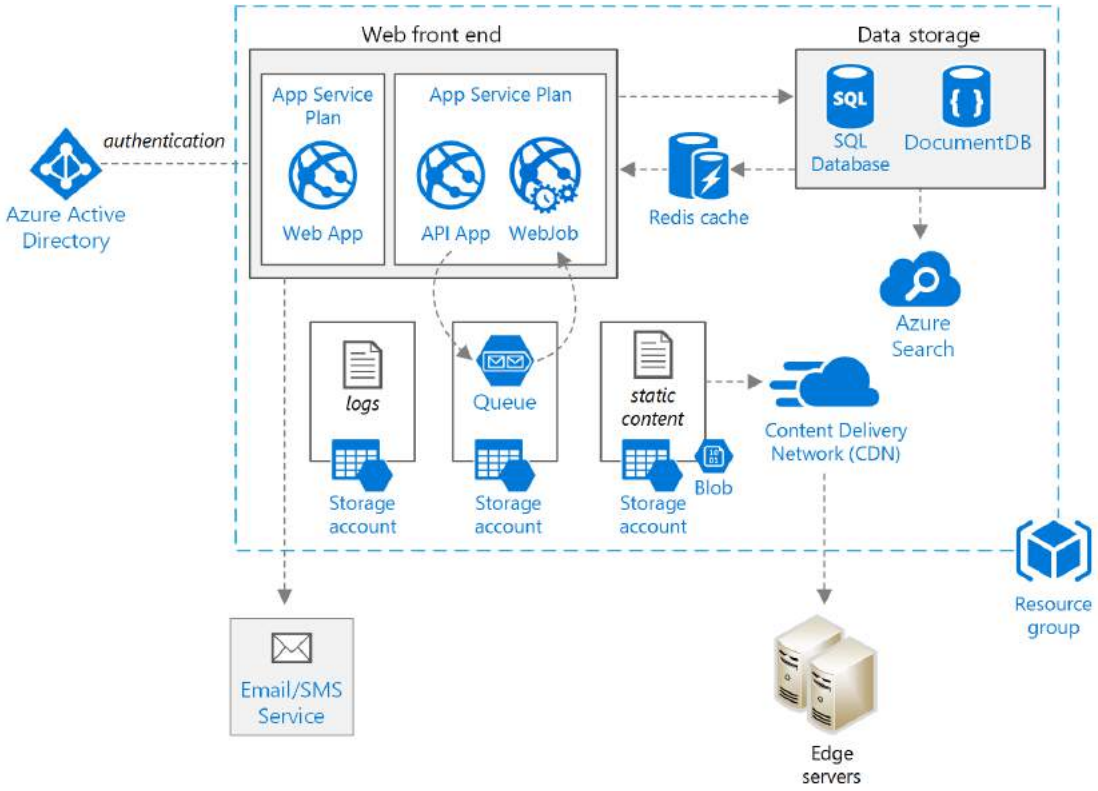

Figura 11. *Arquitectura de una aplicación Web en Microsoft Azure App Service*

Microsoft Azure es el que ofrece menos **período gratuito** [75], ya que únicamente proporciona 30 días y 170€ para realizar pruebas. Considero que el tiempo es demasiado escaso para poder realizar una aplicación completa y realizar las pruebas oportunas para poder valorar la plataforma y el servicio de App Service.
Igual que en las otras plataformas, con Microsoft Azure también se paga por las necesidades y el consumo y encontramos diversos **precios** [76] a los que adaptarnos, por lo que supone una ventaja a nivel económico para el cliente. También nos ofrece una **calculadora de precios** [77] para poder estimar el gasto que supondría albergar la aplicación en App Service.

Microsoft Azure también ofrece mucha **documentación** [78] sobre App Service aunque la mayoría está en español, en algunos casos la información que proporciona es demasiado básica y encontrar información algo más avanzada supone invertir mucho más tiempo, además de que existe mucha más información para aplicaciones ASP.NET que para Java.

El **entorno de la plataforma Microsoft Azure** es sencillo, más similar al de Google Cloud Platform que al de Amazon Web Services ya que el acceso a los servicios los encontramos en un panel lateral y en la parte central aparecerán las diversas opciones del servicio seleccionado. Además, la ubicación de los elementos de la parte central también es personalizable.

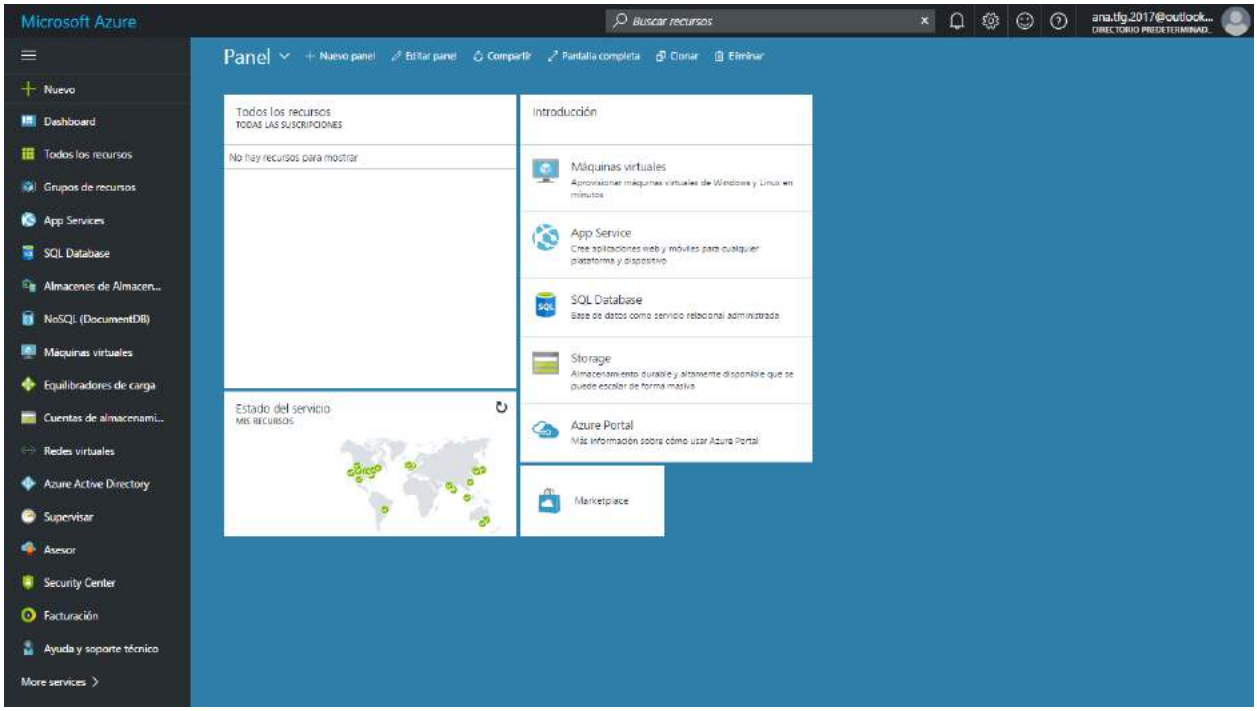

Figura 12. *Captura de pantalla de la plataforma Microsoft Azure.*

Al acceder a **App Service** la primera vez nos encontramos una interfaz algo escueta, ya que inicialmente no tenemos ninguna aplicación creada.

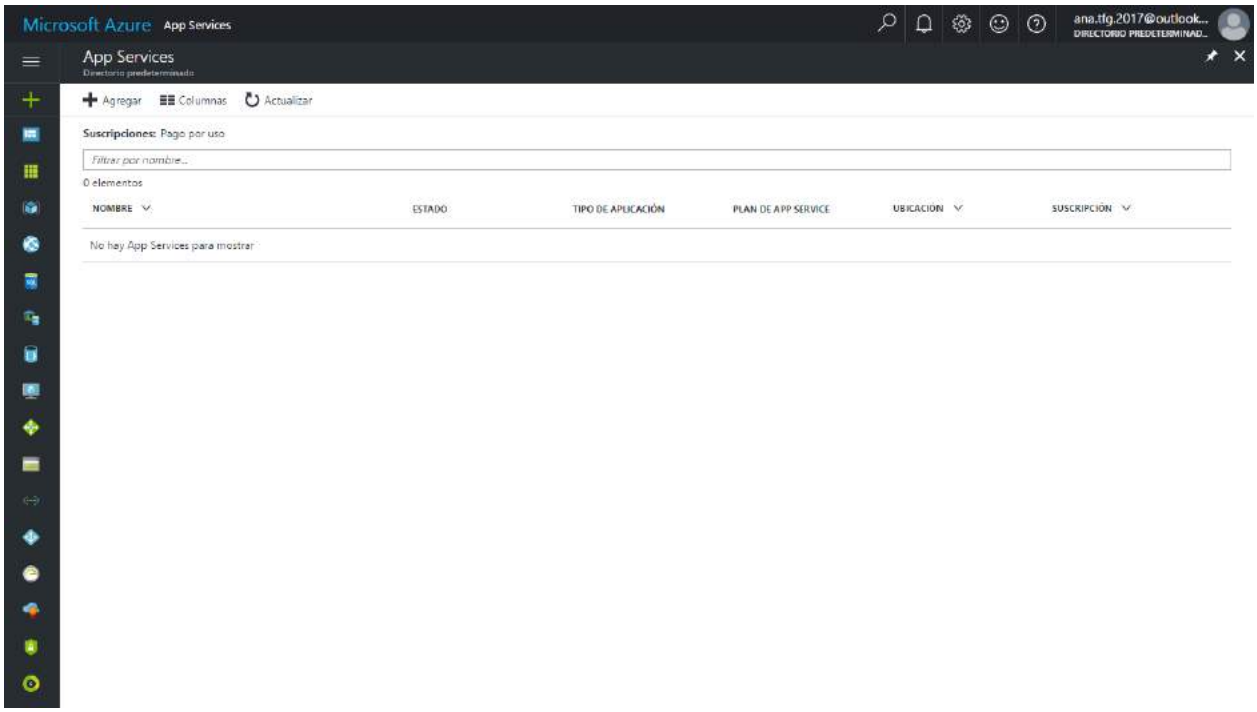

Figura 13. *Captura de pantalla de App Service.*

Al agregar una nueva aplicación podremos elegir entre una gran variedad de plantillas y al seleccionar cualquiera de ellas nos proporciona una breve explicación sobre ella.

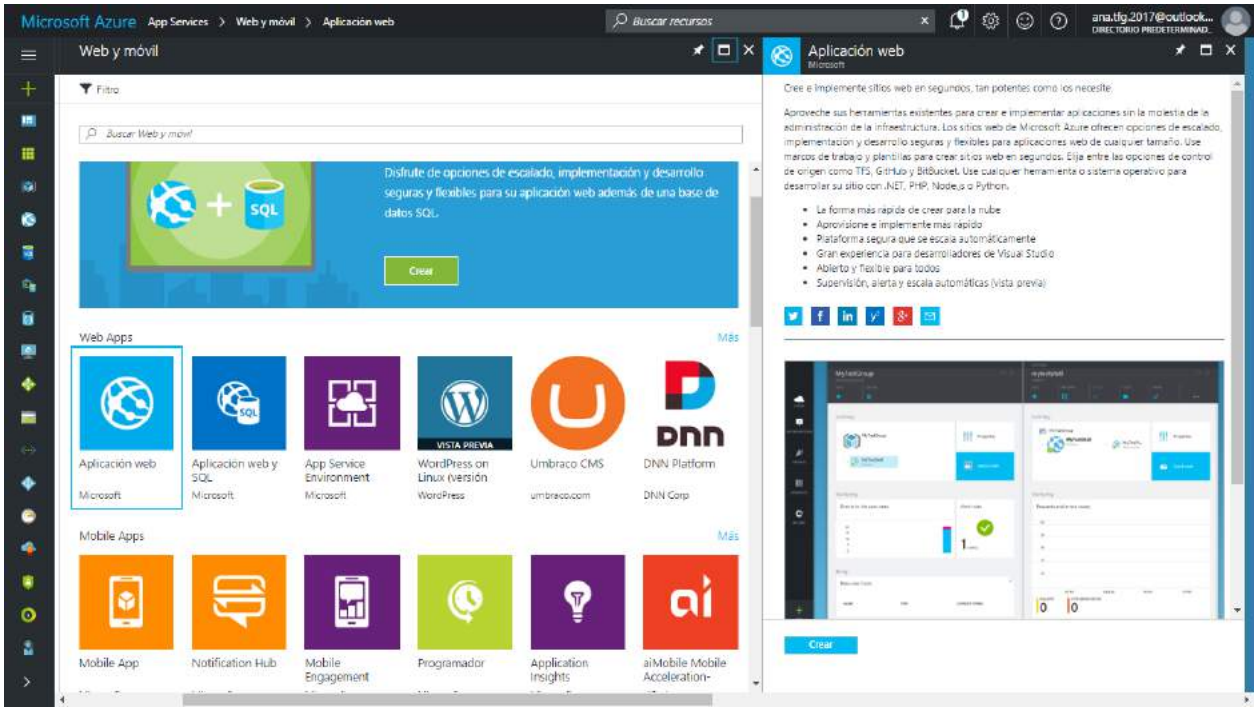

Figura 14. *Captura de pantalla de las plantillas y su explicación en App Service.*

La creación de una aplicación web, ya sea nueva o basada en algún ejemplo, es bastante sencilla, sólo hay que indicar el nombre de la aplicación, la suscripción, el grupo de recursos y el plan de app service y la ubicación.

Una vez creada la aplicación, Microsoft Azure proporciona un **subdominio** [79] de azurewebsites.net, aunque también lo podremos personalizar. En el caso de utilizar el dominio que nos ofrece, nos proporciona la **URL**: http://nombre-de-la-app.azurewebsites.net. Por tanto, igual que ocurre en el caso de Google, dependiendo del nombre de la aplicación, la URL será una más o menos clara y fácil de recordar.

A nivel de desarrollo, también hay que tener en cuenta que podremos utilizar el **plugin** que Microsoft ofrece **para Eclipse** [80], facilitando el trabajo y las pruebas a la hora de implementar el proyecto.

Otra característica diferente a Google es que, al igual que Amazon, podremos crear nuestra aplicación con la **versión 8 de Java**.

Como en los casos anteriores, también hay que ser conscientes de que existe la **posibilidad de un fallo en la infraestructura de Microsoft Azure** que nos dejaría sin servicio.

3.2 Justificación de la propuesta final

Teniendo en cuenta las ventajas e inconvenientes descritas en el apartado anterior, se ha elegido para implementar la aplicación la plataforma de **Google Cloud Platform** con el servicio de **App Engine**. Se elige esta plataforma teniendo en cuenta la arquitectura propuesta, la URL que proporciona, el tiempo disponible de la versión gratuita, la facilidad de uso de la plataforma en sí además de toda la documentación y cursos disponibles que hay relacionada con ella.

A continuación, se pasa a detallar la planificación de recursos, así como el impacto económico.

3.2.1 Planificación de recursos

En este trabajo de final de grado, voy a suponer que la propuesta la desarrollará una empresa dedicada a la implementación de este tipo de soluciones, ubicada en Alcoy.

La **planificación de las fases y tareas**, así como la duración estimada de las mismas y la asignación del personal necesario se muestra en la siguiente figura:

	Nombre de tarea	Duración	Comienzo	Fin	Trabajo	Nombres de los recursos	Predecesoras
1	**ANÁLISIS**				**16 horas**		
2	Estudio del estado del arte	1 día	mar 02/05/17	mar 02/05/17	4 horas	Analista de software	
3	Análisis de requerimientos	1 día	mar 02/05/17	mar 02/05/17	4 horas	Analista de software	2
4	Especificaciones funcionales del sistema	1 día	mié 03/05/17	mié 03/05/17	8 horas	Analista de software	3
5	**DISEÑO**				**24 horas**		
6	Diseño de la arquitectura	1 día	jue 04/05/17	jue 04/05/17	4 horas	Diseñador de bases de datos	4
7	Diseño del modelo de datos	1 día	jue 04/05/17	jue 04/05/17	4 horas	Diseñador de bases de datos	4
8	Diseño de la interfaz	2 días	jue 04/05/17	vie 05/05/17	16 horas	Diseñador web	4
9	**IMPLEMENTACIÓN**				**120 horas**		
10	Implementación backend	10 días	vie 05/05/17	jue 18/05/17	80 horas	Desarrollador Java	6;7
11	Implementación frontend	5 días	lun 08/05/17	vie 12/05/17	40 horas	Programador web	8
12	**PRUEBAS**				**6 horas**		
13	Testeo de la aplicación	1 día	vie 19/05/17	vie 19/05/17	6 horas	Soporte técnico	10;11
14	**DOCUMENTACIÓN**				**24 horas**		
15	Manual técnico	14 días	mar 02/05/17	vie 19/05/17	20 horas	Todos	
16	Manual de usuario	1 día	lun 22/05/17	lun 22/05/17	4 horas	Soporte técnico	13

Figura 15. *Detalle de fases y tareas del proyecto.*

Suponiendo que el proyecto empezara a principio de Mayo, finalizaría el mismo mes, tal y como se muestra en el siguiente diagrama de Gantt:

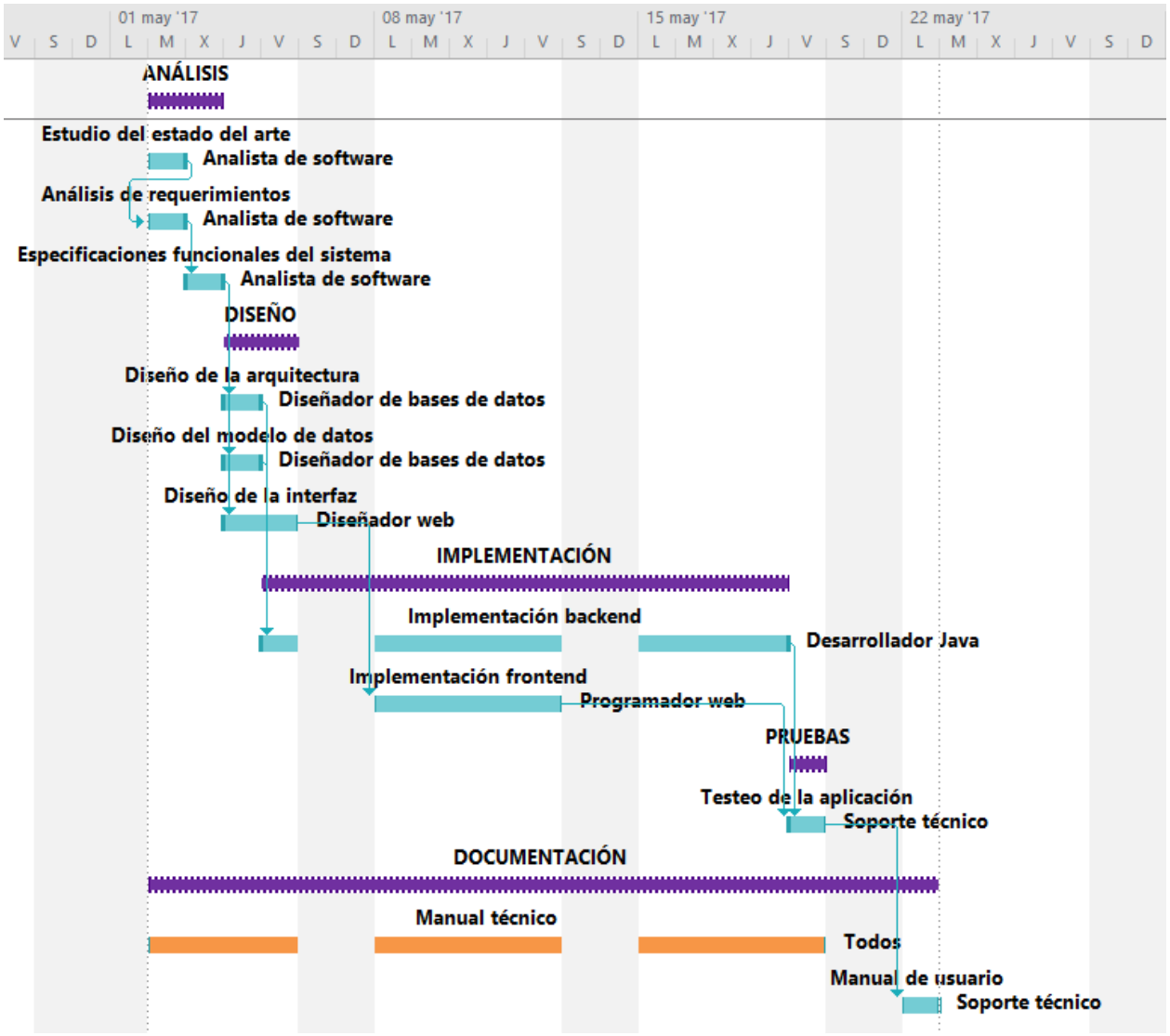

Figura 16. *Diagrama de Gantt de la planificación del proyecto.*

Para finalizar este punto, quiero comentar que las fases y las tareas para realizar este proyecto serán las mismas, tanto si el proyecto lo realiza una empresa especializada en este tipo de desarrollos como en este caso en el que el desarrollo ha sido realizado en su totalidad por mí, pero la diferencia más notable serán las horas de trabajo dedicadas ya que en cuanto al estudio del estado del arte y el llevar a cabo todo el diseño e implementación me ha llevado más del triple de tiempo, sin contar las horas no planificadas para poder conseguir el funcionamiento correcto de la aplicación.

En el siguiente punto, realizo una simulación del impacto económico que supondría al cliente el llevar a cabo la puesta en marcha de este proyecto a través de la empresa especializada en este tipo de desarrollos.

3.2.2 Impacto económico

Inicialmente, haciendo uso de una cuenta gratuita durante el primer año, el cliente únicamente deberá abonar los **costes de las horas de trabajo** [81], tal y como se detalla a continuación:

COSTES DE LAS HORAS DE TRABAJO			
PERSONAL	**COSTE/HORA (€)**	**Nº DE HORAS**	**TOTAL (€)**
Analista de software	25,00 €	16	400,00 €
Diseñador bases de datos	25,00 €	8	200,00 €
Diseñador web	25,00 €	16	400,00 €
Desarrollador Java	20,00 €	80	1.600,00 €
Programador web	20,00 €	40	800,00 €
Soporte técnico	15,00 €	10	150,00 €
		BASE IMPONIBLE	3.550,00 €
		IVA (21%)	745,50 €
		TOTAL	**4.295,50 €**

Como el precio de App Engine se determina dependiendo del consumo e inicialmente no sabemos el impacto que va a tener la aplicación voy a hacer una **estimación mínima del coste mensual** [47], a partir de ahí se iría incrementando el precio.

GASTOS MENSUALES DE APP ENGINE

CONCEPTO	CANTIDAD/UNIDAD	COSTE	TOTAL (€)
INSTANCIA B1			
128 MB de RAM y 600 MHz de CPU	COSTE POR HORA	0,05 €	32,10 €
GOOGLE CLOUD DATASTORE			
Datos almacenados	1 GB AL DÍA	0,00 €	0,00 €
Lectura de entidades	50.000 AL DÍA	0,00 €	0,00 €
Escritura de entidades	20.000 AL DÍA	0,00 €	0,00 €
Eliminación de entidades	20.000 AL DÍA	0,00 €	0,00 €
API DE BÚSQUEDAS			
Almacenamiento Total (Documentos e índices)	0,25 GB	0,00 €	0,00 €
Consultas	1.000 AL DÍA	0,00 €	0,00 €
Documentos de búsquedas de indexación	0,01 GB AL DÍA	0,00 €	0,00 €
OTROS RECURSOS			
Tráfico de red - saliente	1 GB	0,11 €	0,11 €
Tráfico de red - entrante	1 GB	0,00 €	0,00 €
API de registros	1 GB	0,11 €	0,11 €
OTROS RECURSOS SIN CARGO ADICIONAL			
Memcaché compartida	-	0,00 €	0,00 €
Envío de emails	-	0,00 €	0,00 €
Cron	-	0,00 €	0,00 €
API(extracción de URL, colas de tareas, imagen, sockets, archivos, usuarios, y canal)	-	0,00 €	0,00 €
		SUBTOTAL	32,32 €
		IVA (21%)	6,79 €
		TOTAL	**39,11 €**

4. Implementación

4.1 Entorno de desarrollo

El **IDE para el desarrollo** de la aplicación utilizado ha sido **Eclipse Mars 2** por su compatibilidad con Java 7, Apache Maven y App Engine.

Para llevar a cabo la implementación de la aplicación web se ha instalado previamente el **JDK de Java**. También se ha procedido a instalar **Apache Maven** en su versión 3.3.9 y el **SDK de Google Cloud** [59]. Una vez instalado el SDK de Google Cloud, a través de la consola de comandos se agrega el componente gcloud que incluye el **SDK de App Engine para Java**.

Finalmente, se ha configurado el entorno de desarrollo de Eclipse Mars instalando el **plugin Google Cloud Tools for Eclipse 1.2.0.**

Para la **implementación de la aplicación** se han utilizado [57][82][83]:

- **Archivos Java:** para las entidades de la aplicación (perfil, mascota y visita), para configurar el servicio Objectify personalizado que utilizará la aplicación. También las clases simple Java (POJO) se utilizan para la representación de formularios y filtros. Así como para crear las servlets para el envío de emails y para crear la API general de toda nuestra aplicación.
- **AngularJS:** para poder codificar todos los controles que determinan todo el funcionamiento de las páginas web.
- **Páginas web HTML con CSS y Bootstrap:** para el diseño responsive de todo el front-end de la aplicación web.
- **Archivos XML:** para la configuración de la aplicación, en este caso, el archivo *pom.xml* contiene la configuración del proyecto Maven con todas las dependencias necesarias, el archivo *web.xml* donde aparece la configuración del proyecto web, el archivo *web-appengine.xml* con la configuración del proyecto para App Engine y finalmente el archivo *datastore-indexes.xml* donde se incluirán todos los índices para poder realizar posteriormente los filtros en los datos del datastore.

Por otra parte, para poder llevar a cabo la actualización de la aplicación en App Engine desde Eclipse simplemente hay que especificar la **configuración de ejecución de la aplicación** tal y como muestra la siguiente imagen:

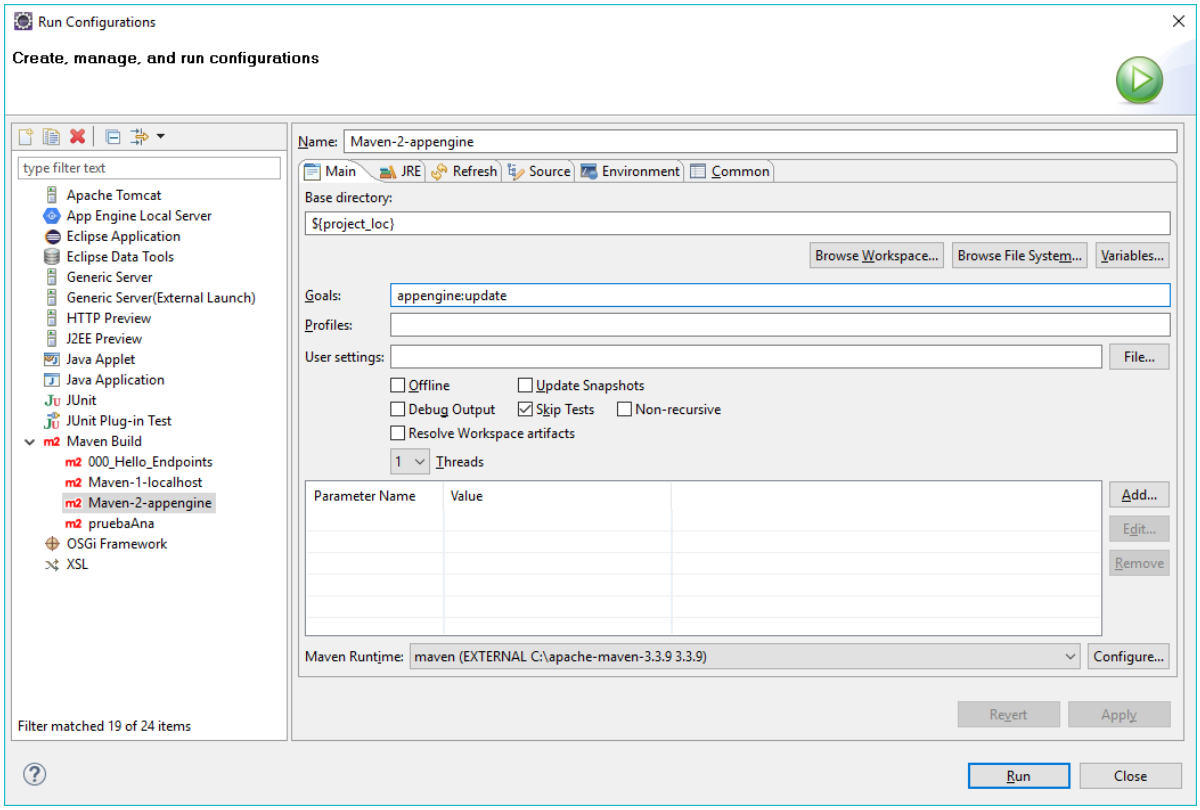

Figura 17. *Configuración del entorno de ejecución para actualizar la aplicación en App Engine.*

Para finalizar este apartado, hay que tener en cuenta que actualizando nuestra aplicación aparezca un error porque alguna transacción anterior se haya quedado en progreso, esto provoca un conflicto y nos impide actualizar la aplicación. Para solucionar esto, deberemos realizar un rollback de nuestros proyectos; para ello accederemos a la consola de comandos de Windows y dentro del directorio donde tenemos nuestro espacio de trabajo de Eclipse ejecutaremos el siguiente comando: `mvn appengine:rollback` de esta forma se cancelará cualquier actualización anterior solucionando cualquier conflicto.

4.2 Implementación práctica

4.2.1 Diseño del modelo de datos

App Engine puede trabajar con varias tecnologías de almacenamiento pero, en este caso, se ha elegido **Datastore** [84] [85].

Datastore es una base de datos que se ejecuta en la nube de Google, disponible para cualquier aplicación de App Engine. Está construido sobre BigTable [86], la cual es una tecnología creada por Google que permite almacenar y buscar con efectividad una gran cantidad de datos y por ello es la usada por muchos de los grandes servicios de Google, como el motor de búsquedas Google Search, el servidor de correo electrónico Gmail y la herramienta de analítica web Google Analytics.

Datastore es una base de datos NoSQL, concretamente es un **repositorio "clave-valor" orientado a columnas**.

- El concepto "NoSQL" determina que no tiene una estructura fija tal y como se conoce en las bases de datos relacionales e incluso permite tener propiedades heterogéneas y multivalor en una misma entidad. Aunque se pueda utilizar JDO y JPA no significa que sea relacional.
- El concepto "clave-valor" es muy similar a las tablas hash o arrays asociativos de los diversos lenguajes de programación y determina la forma en que obtiene y almacena los datos.
- El concepto "orientado a columnas" es porque las columnas, y no las filas, se almacenan juntas, lo que proporciona un mejor rendimiento y escalabilidad.

Otras características [87] de Dastore son:

- Es transaccional y tiene las propiedades ACID por lo que proporciona persistencia.
- Se particiona de forma nativa porque es un sistema distribuido y sus datos se reparten entre distintos servidores. Incluso en centro de datos independientes, esta distribución aumente el rendimiento porque proporciona la posibilidad de recuperar datos de varios lugares en paralelo.

Todas estas características proporcionan a nuestra aplicación escalabilidad, rendimiento y replicación sin tener que hacer nada extra en la aplicación, simplemente hay que diseñar correctamente el modelo de datos.

Por otra parte, hay que tener en cuenta los **principios de modelo del almacenamiento** de datos en el Datastore: [84]

- TIPO (*Kind*): define una estructura particular, a partir de un tipo se crearán las entidades.
- ENTIDAD (*Entity*): define los objetos en el datastore. Cada entidad tendrá una o varias propiedades y cada una de ellas podrá tener uno o más valores, incluso las entidades del mismo tipo podrán tener diferentes propiedades y el tipo de datos de cada una de esas propiedades no tiene porqué ser el mismo.
- PROPIEDAD (*Property*): definirá las características de la entidad.

La siguiente tabla muestra la r**elación entre las diferentes tecnologías y su terminología**:

TECNOLOGÍA	ESTRUCTURA	INSTANCIAS	PROPIEDADES
Datastore	TIPO	ENTIDAD	PROPIEDAD
Programación Orientada a Objetos	CLASE	OBJETO	CAMPO/ATRIBUTO
Bases de datos relacionales	TABLA	FILA	COLUMNA

Hay que tener en cuenta que cada entidad deberá tener una **clave única** que será la forma de identificarla, y hay que tener en cuenta que habrá dos formas de generar dicha clave:

1. De forma automática: en este caso será el Datastore quién genere esta clave. Posteriormente podremos recuperar la entidad obteniendo su valor
2. De forma explícita: se creará una propiedad en la entidad que será su identidad y el almacén de datos utilizará dicho valor para generar la clave. Esta forma nos proporciona la ventaja de que podremos recuperar la entidad usando dicho valor directamente.

En cuanto a las **relaciones** existentes entre las entidades encontramos dos tipos:

- Antecesor: nos permitirá establecer una estructura jerárquica entre las entidades. Este tipo de relación debe asignarse al crear la entidad y nunca podrá modificarse.
- HAS-A: no tiene las reglas estrictas de las relaciones antecesor y podrán ser modificadas.

En mi aplicación la entidad perfil, la cual almacena los datos del perfil del usuario, tendrá una clave única creada de forma explícita mientras que las entidades mascota y visita, las cuales almacenan los datos de las mascotas y de sus visitas respectivamente, tendrán una clave única creada de forma automática. Por lo que representa a las relaciones, he decidido crear una relación antecesor entre la entidad perfil y la entidad mascota, y entre la entidad perfil y la entidad visitas, ya que en ambos casos el usuario es el creador, tanto de mascotas como de sus visitas.

Teniendo en cuenta todos los puntos relacionados con el modelo de datos, a continuación, se representa de forma gráfica el modelo de datos inicial diseñado:

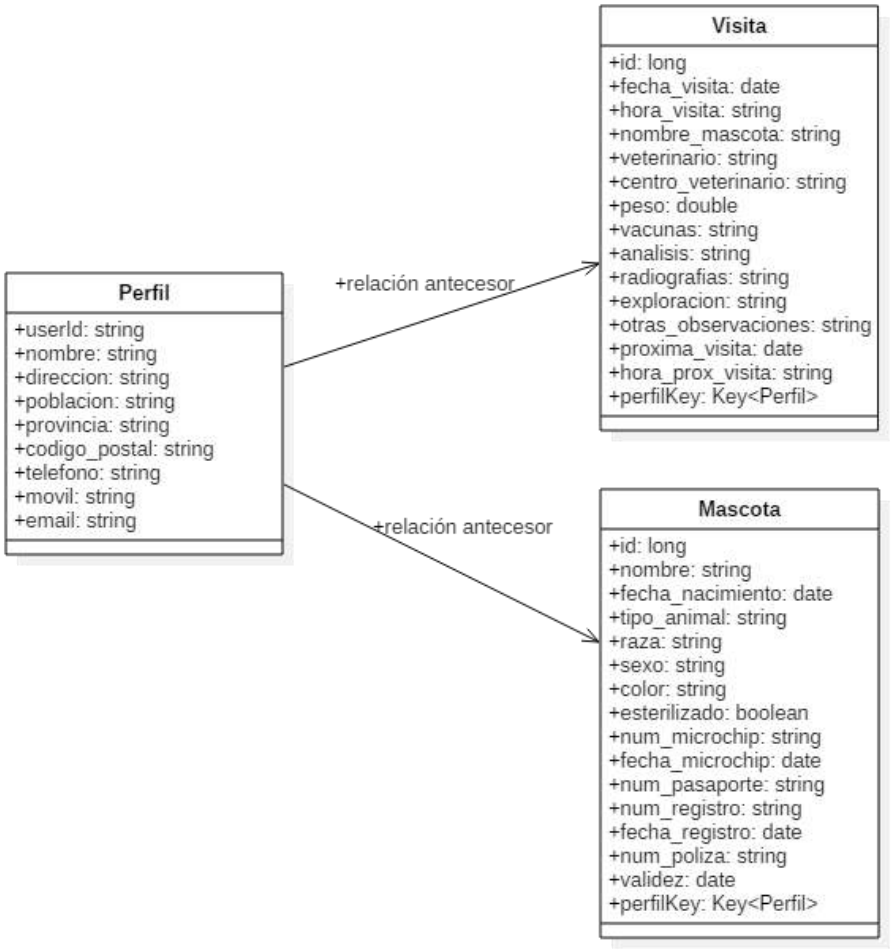

***Figura 18**. Modelo de datos y relaciones*

4.2.2 Maquetas

A continuación, se muestran los **diseños iniciales de la aplicación**, aunque en algún caso se han llevado a cabo diversos ajustes para que al acceder desde diferentes dispositivos permita al usuario navegar correctamente por la aplicación:

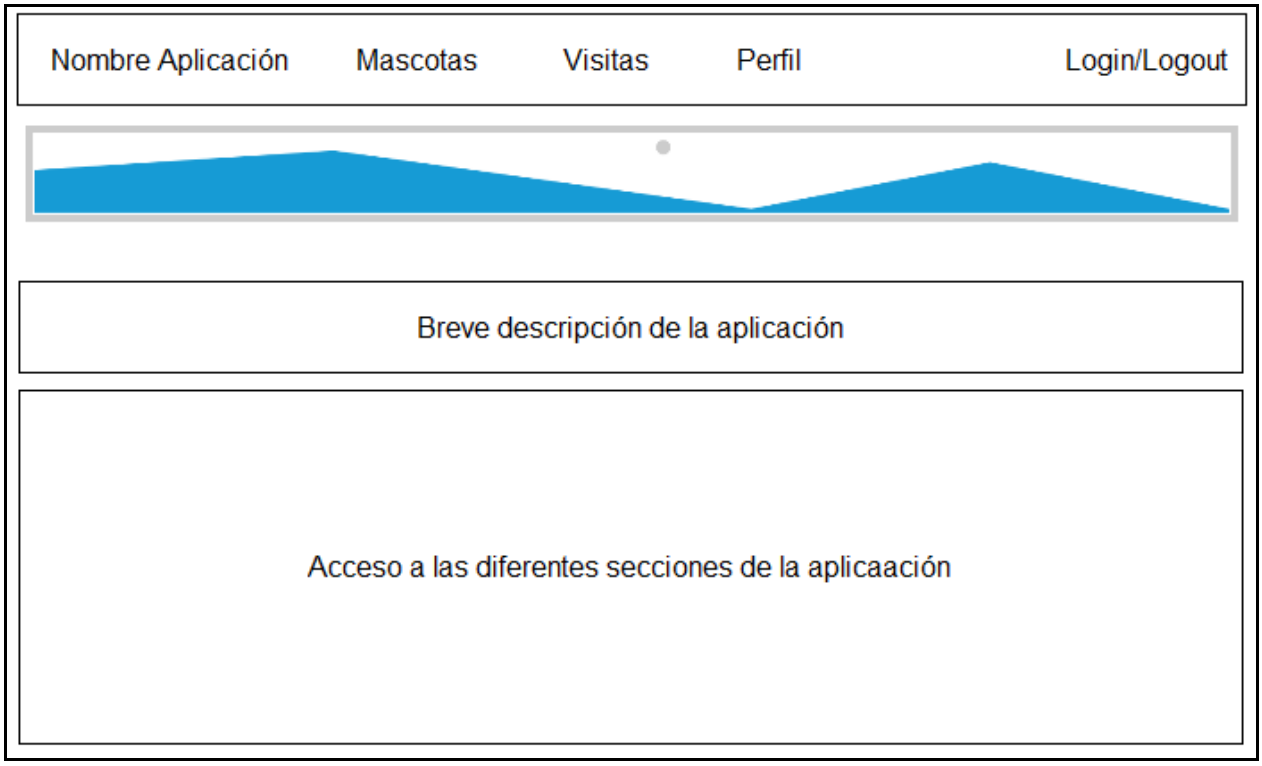

Figura 19. *Página principal*

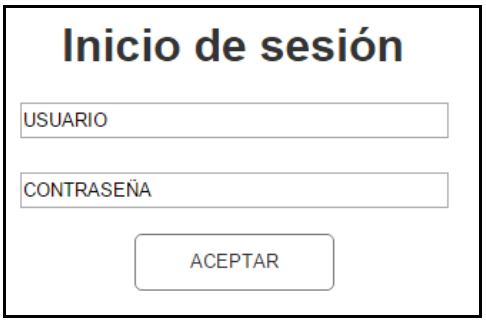

Figura 20. *Página de inicio de sesión (login)*

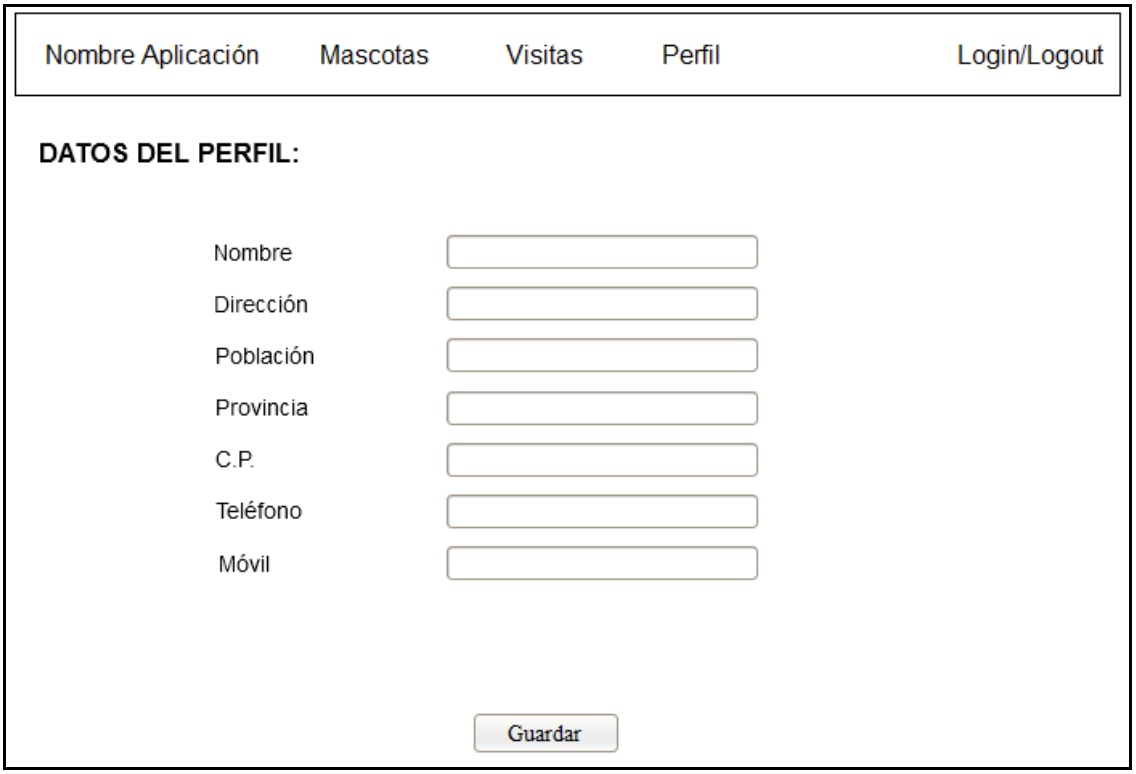

Figura 21. *Página del perfil de usuario*

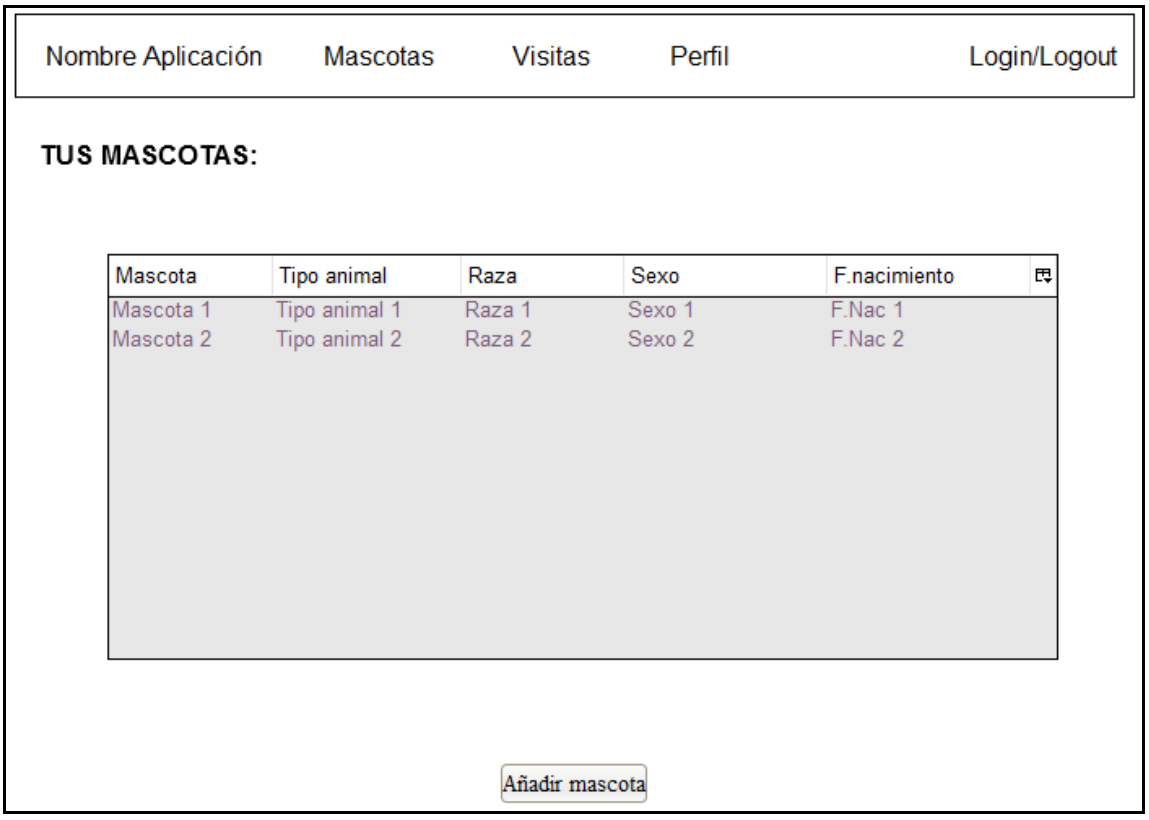

Figura 22. *Página del listado de mascotas*

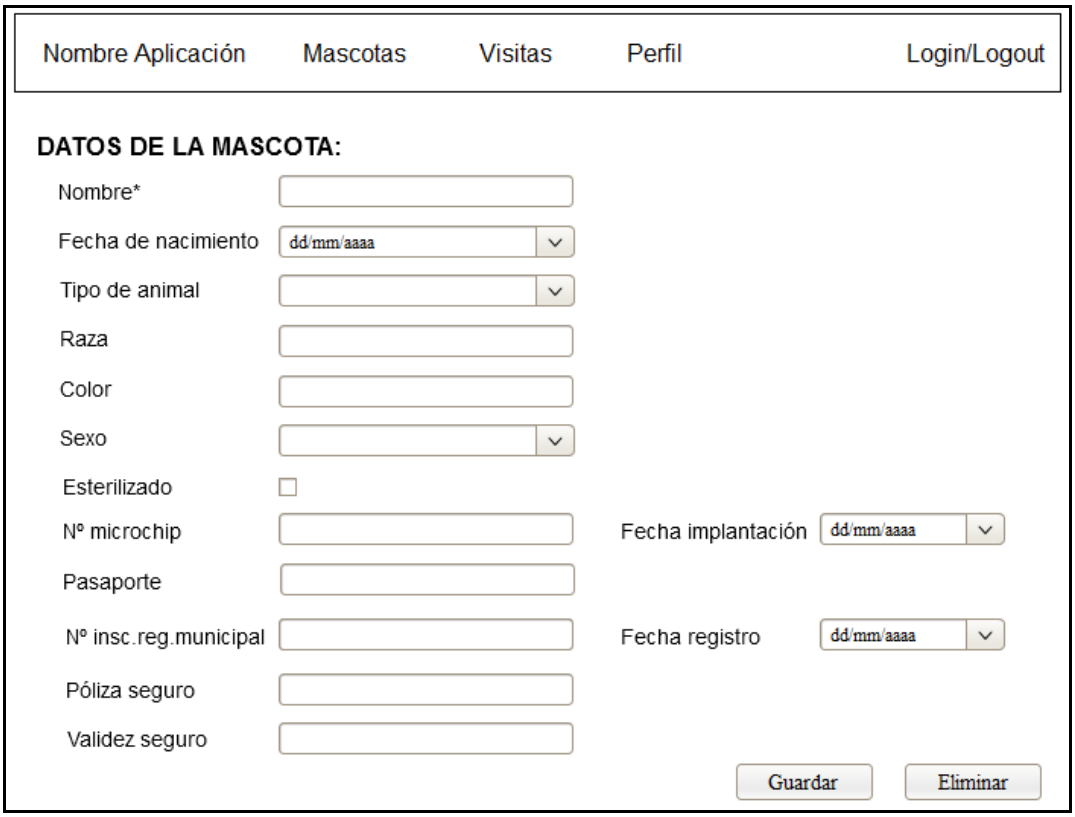

Figura 23. *Página para crear nuevas mascotas o editar los datos*

Figura 24. *Página del listado de visitas*

Figura 25. *Página para crear nuevas visitas o editar los datos*

4.2.3 Diagramas de funcionamiento

En este apartado, se presenta el **diagrama de funcionamiento**:

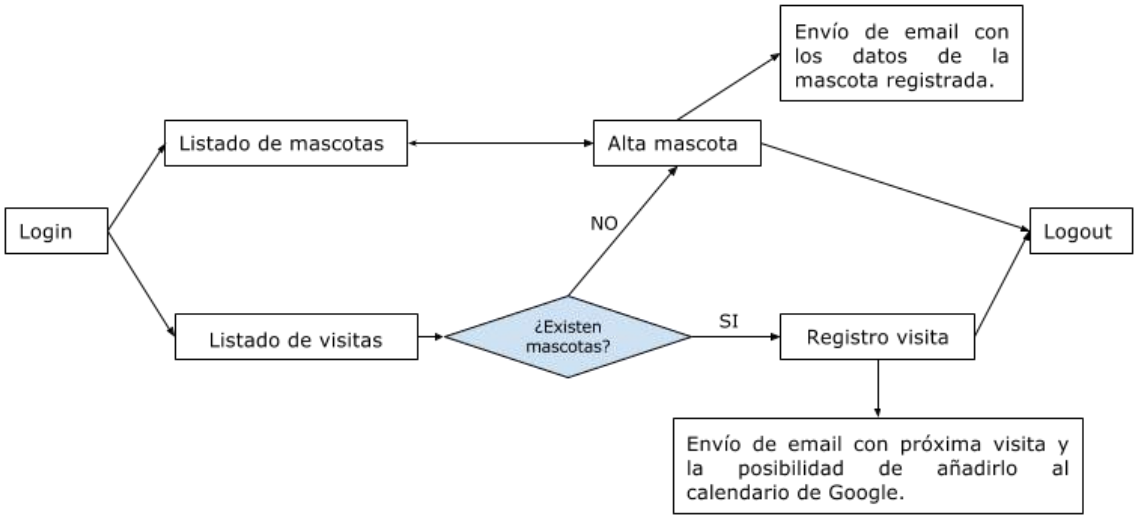

Figura 26. *Diagrama de funcionamiento de la aplicación web.*

4.3 Pruebas

4.3.1 De sistema

En primer lugar, una vez implementada la **API** de la aplicación se ha procedido a realizar las **pruebas de funcionamiento** correspondientes. Para acceder al explorador de la API hay que acceder a la siguiente URL: https://centro-veterinario.appspot.com/_ah/api/explorer

Desde ahí una vez seleccionado "veterinario API" podremos visualizar todos los métodos implementados en la API para la aplicación del veterinario.

Se ha realizado una prueba para cada uno de los métodos para comprobar su funcionamiento.

Una vez comprobado el funcionamiento de la API e introducido a través de ella diversas entidades, **se comprueba que las entidades han sido almacenadas en el datastore**, para ello se accede al proyecto desde la plataforma de Google Cloud (https://console.cloud.google.com) y dentro del datastore se comprueban que aparecen dichas entidades. Desde aquí se pueden realizar consultas GQL [88] para poder buscar entidades.

Una consulta GQL es similar a una SQL pero en este caso en vez del nombre de la tabla que se utiliza en SQL se debe indicar el nombre de la entidad, por ejemplo:

```
Select * FROM Mascota
WHERE nombre='Kiro'
```

Devolvería todos los datos de la mascota que corresponda con dicho nombre. Las consultas GQL son para una determinada entidad, por lo que no podríamos realizar consultas que influyeran en varias entidades.

Finalmente, se ha procedido a **comprobar el acceso a la Web** desde el navegador utilizando la URL: https://centro-veterinario.appspot.com

4.3.2 De integración de sistemas

Una vez se han realizado las pruebas de sistema se ha pasado a **testear el funcionamiento completo de la aplicación web**.

Se ha accedido a la URL: https://centro-veterinario.appspot.com, se ha probado el control de inicio de sesión desde todas las opciones. Se ha iniciado la sesión y se ha configurado el perfil, se han añadido/actualizado/eliminado mascotas y visitas, también se ha comprobado la recepción de los correspondientes emails y se ha procedido a realizar el filtrado de datos para comprobar que todo funciona de la manera esperada.

4.3.3 De volumen

Al trabajar con una **cuenta gratuita** de Google Cloud Platform, y como ya hemos ido comentado, tendremos una serie de **limitaciones** a la hora de explotar la aplicación web [89].

El principal aspecto a tener en cuenta es que desde esta versión de prueba no podemos configurar el tipo de instancia [13] en el que se ejecutará la aplicación por lo que deberemos tener en cuenta que podremos encontrar cierta lentitud de la aplicación mientras no pasemos a un tipo de cuenta de facturación y también hay que tener en cuenta el tipo de conexión a Internet con la que se conecte el usuario.

Otro punto a tener en cuenta es la escalabilidad, en este caso, como se ha comentado en el punto 4.2.1 se ha optado por almacenar los datos en datastore por la escalabilidad, pero no podremos beneficiarnos de una escalabilidad automática y balanceo de carga a menos que pasemos a un cuenta de facturación.

5. Resultados

5.1 Migración al entorno de producción

En este caso la aplicación web del centro veterinario **ya está en un entorno de producción** ya que es pública para cualquier usuario a través de la URL https://centro-veterinario.appspot.com
Hay que tener en cuenta que al haberse implementado bajo una cuenta gratuita de Google Cloud tiene una serie de limitaciones [48] que serán más notables cuando el número de usuarios vaya aumentando, por ejemplo, la limitación existente en el envío de email diarios que en este caso es de 10 emails al día. Este tipo de limitación se podrá eliminar pasando a una cuenta de pago de Google Cloud [47].

5.2 Manual de usuario

Al tratarse de una **aplicación web** no necesita **ningún tipo de instalación** por parte del usuario, únicamente deberán tener acceso a un navegador web y una conexión a Internet.

Además, posee un **diseño responsive**, el cual hace que la aplicación web se visualice correctamente desde cualquier dispositivo.

Figura 27A. *Captura de una parte de la página principal de la aplicación web visto desde un navegador en un PC.*

Figura 27B. *Captura de una parte de la página principal de la aplicación web visto desde el navegador de un móvil.*

Para cumplir con la legalidad vigente en cuanto a las cookies y a la política de privacidad, en la **parte inferior** de la página aparecerá el **aviso del uso de cookies**, y además en el pie de todas las páginas de la aplicación se encontrará el vínculo a la **política de privacidad**.

5.2.1 Explotación

La aplicación es de **uso** muy **sencillo** e **intuitivo** para el usuario.

El usuario deberá **acceder a la URL**: https://centro-veterinario.appspot.com donde le aparecerá la página principal con una barra de navegación en la parte superior de la pantalla, una breve explicación del servicio y a continuación el acceso a todas las secciones de la aplicación para gestionar las mascotas, las visitas y el perfil de usuario.

Una vez cargada la aplicación, y siguiendo el diagrama de funcionamiento presentado en el anterior punto 4.2.3:

1. El usuario deberá **iniciar su sesión con una cuenta de Google**, pulsando sobre el botón "Iniciar sesión" situado en la parte superior derecha. También se solicitará el inicio de sesión cuando se intente acceder a cualquier sección sin haberse logueado previamente.

   Para poder iniciar la sesión el usuario deberá aceptar la política de privacidad del Centro Veterinario. Una vez aceptada, aparecerá la ventana de autenticación de la aplicación en la que el usuario deberá introducir su dirección de correo electrónico y su correspondiente contraseña.

2. La primera vez que se registra el usuario sería recomendable que se rellenaran los **datos del perfil**, desde la opción "Perfil" situada en la barra de navegación superior o desde el botón "Acceder al perfil" que se encuentra el apartado "Actualiza tu perfil de usuario" de la página principal del centro veterinario.

   Desde la ventana del formulario, el usuario únicamente deberá introducir los datos, aceptar la política de privacidad y pulsar el botón "Actualizar perfil" (situado en la parte inferior del formulario).

3. Para llevar a cabo la **gestión de las mascotas** se pulsará sobre la opción "Mascotas" situada en la barra de navegación superior o desde el botón "Ver mascotas" que se encuentra el apartado "Tus mascotas" de la página principal del centro veterinario. Desde esta sección el usuario podrá visualizar el **listado de todas sus mascotas** cuando las haya dado de alta.

   Para **crear nuevas mascotas**, se pulsará el botón "Dar de alta mascotas", en este momento se cargará el formulario para introducir todos los datos de la mascota. Obligatoriamente se tiene que introducir el nombre de la mascota, ya que es un campo requerido, todos los demás se podrá introducir, si se desea, más adelante. Cuando hayamos introducido todos los datos deseados, pulsaremos el botón "Dar de alta" situado en la parte inferior del formulario y la aplicación nos redirigirá al listado de mascotas, de esta forma podremos comprobar que la mascota se ha almacenado correctamente.

   Cada vez que se registre una mascota, automáticamente, el usuario **recibirá** en su correo un **email** con los datos de la mascota que ha dado de alta.

   Volviendo al listado de mascotas, al pulsar en el hipervínculo "Detalles" que hay delante del nombre de nuestra mascota, se cargará el formulario con todos los datos de la mascota, desde ahí se podrá **actualizar** cualquier **información**, excepto el nombre. En el momento que se haya actualizado la información, se pulsará el botón "Actualizar" para que se almacene la nueva información. Si lo que se desea es **borrar la mascota** de la aplicación, pulsar el botón "Eliminar".

En los casos en los que los usuarios tengan muchas mascotas, podrán **filtrar el listado de las mascotas** para poder encontrar rápidamente a la mascota deseada. Para ello, se pulsará el botón "Filtro" que nos servirá para añadir el filtro deseado y se rellenarán los datos solicitados: indicar el campo por el que se desea buscar (en este momento, únicamente se puede filtrar por el nombre de la mascota), indicar el operador (el igual = o distinto !=) e introducir el valor buscado. Una vez introducidos los datos pulsar sobre el botón " Buscar". Si existen datos almacenados que se correspondan con la búsqueda introducida, aparecerán dichos registros en pantalla.

Para eliminar el filtro aplicado pulsaremos el botón " Filtros" y, además de eliminar los filtros, volverá a mostrarnos todo el listado de mascotas.

4. Una vez dadas de alta las mascotas, se puede llevar a cabo la **gestión de las visitas al veterinario** desde la opción "Visitas" situada en la barra de navegación superior o desde el botón "Ver visitas" que se encuentra el apartado "Tus visitas" de la página principal del centro veterinario.

   Desde esta sección el usuario podrá visualizar el **listado de todas las visitas** que haya registrado.

   Para **crear nuevas visitas**, se pulsará el botón "Registrar nueva visita", en este momento se cargará el formulario para introducir todos los datos de la visita en cuestión siempre que se haya dado de alta previamente alguna mascota, sino avisará y nos dará la opción de acceder a la creación de mascotas. Obligatoriamente se tiene que introducir la fecha y la hora de la visita, así como el nombre de la mascota, ya que son campos requeridos.

   Cuando se hayan introducido todos los datos deseados, se aceptará la política de privacidad y se pulsará el botón "Registrar" situado en la parte inferior del formulario y la aplicación nos redirigirá al listado de visitas, de esta forma se puede comprobar que la visita se ha almacenado correctamente.

   Cada vez que se registre una visita y se incluya la fecha de la próxima visita, automáticamente, el usuario **recibirá** en su correo un **email** con la fecha y la hora de la **próxima visita** y además un **botón** desde el cual el usuario podrá **añadir** dicha **cita** a su **calendario de Google**.

   De vuelta al listado de visitas, si se pulsa en el hipervínculo "Detalles" que hay delante del nombre de la mascota, accederemos al formulario con todos los datos de la visita, desde ahí se podrá **actualizar** cualquier **información**, excepto la fecha y la hora de la visita y el nombre de la mascota. En el momento que hayamos actualizado la información, se pulsará el botón "Actualizar" para que se almacene la nueva información. Si lo que se desea es **borrar la visita** de la aplicación, pulsar el botón "Eliminar".

   En los casos en los que los usuarios tengan muchas visitas, se podrá **filtrar el listado** para poder encontrar rápidamente la visita deseada. El funcionamiento es exactamente igual que en el caso de las mascotas, detallado previamente, pero en este caso el usuario puede filtrar los datos por el nombre de la mascota y/o por el veterinario que lo atendió.

5. Finalmente, para **cerrar sesión** se pulsa sobre el botón "Cerrar sesión" situado en la parte superior derecha.

5.3 Estadísticas de explotación

Una de las ventajas de utilizar la **plataforma de Google Cloud** es que nos **muestra** diversa **información** sobre la **explotación** de la aplicación, a continuación, se muestran alguna de la información que proporciona:

- Desde el **Panel de Control de App Engine**, podremos visualizar de forma gráfica diversos aspectos de la explotación de nuestra aplicación:

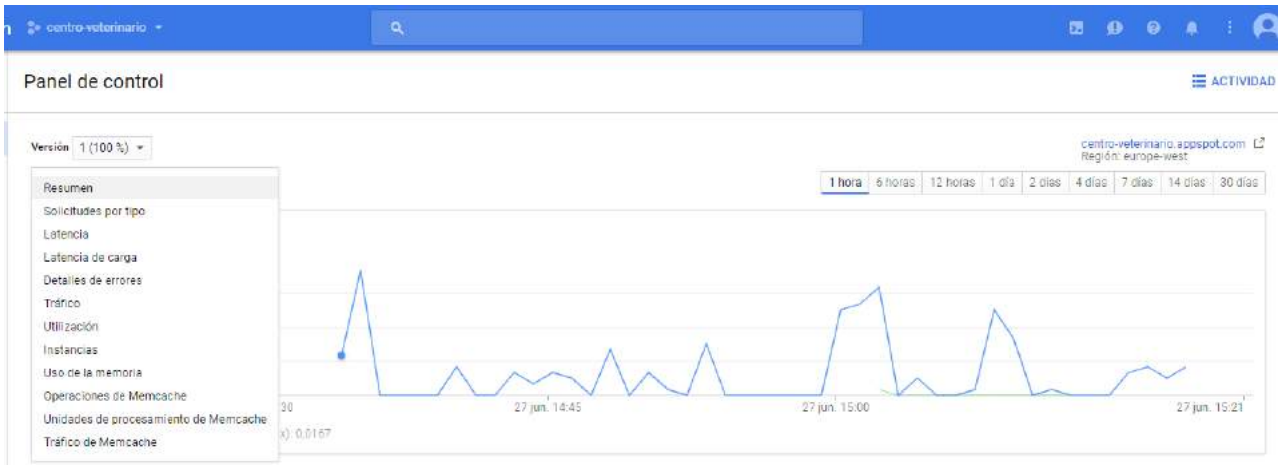

Figura 28. *Captura del gráfico resumen de la última hora y de las opciones que podemos visualizar en el panel de control.*

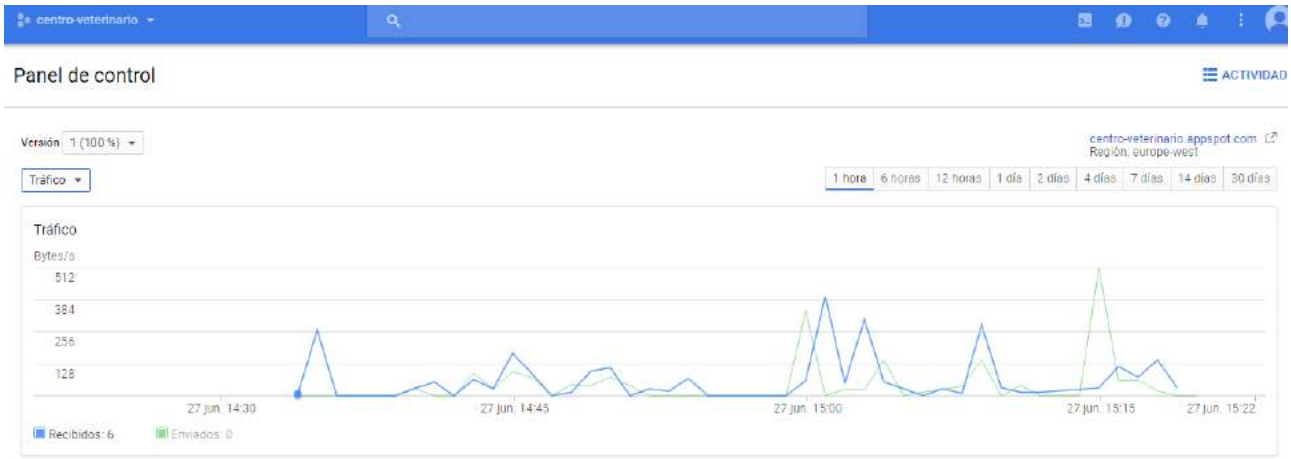

Figura 29. *Captura del gráfico del tráfico de la última hora del panel de control.*

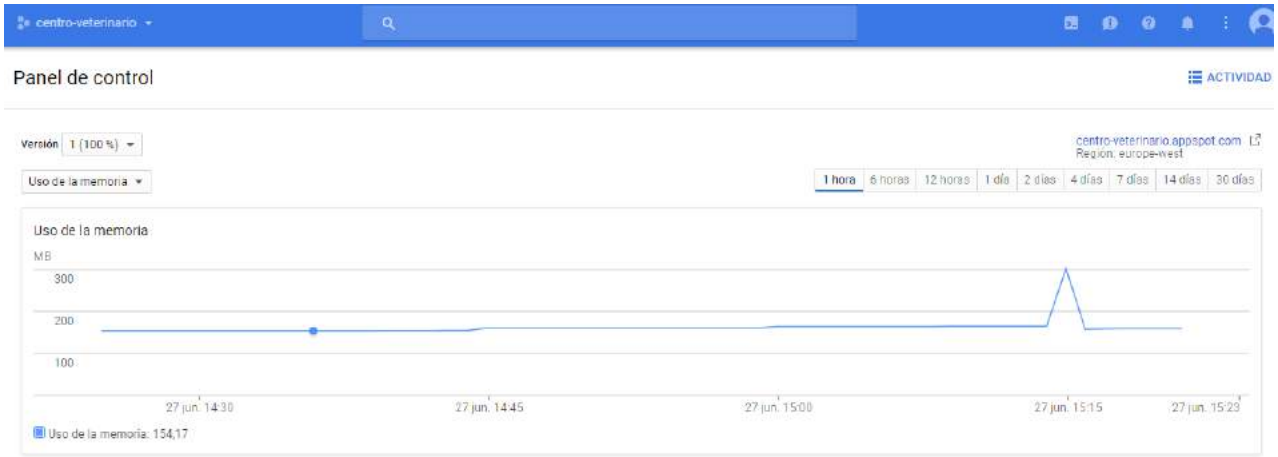

Figura 30. *Captura del gráfico del uso de memoria de la última hora del panel de control.*

- Desde la **Cola de Tareas de App Engine**, podremos visualizar si existen tareas pendientes de ejecutarse y gestionarlas:

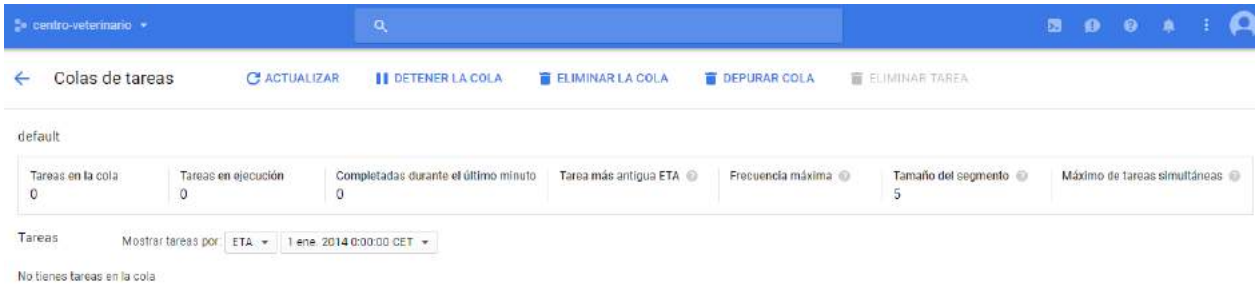

Figura 31. *Captura de la cola de tareas de App Engine.*

- Desde la opción de **Cuotas de App Engine**, podremos controlar el uso de todos los servicios que utiliza nuestra aplicación:

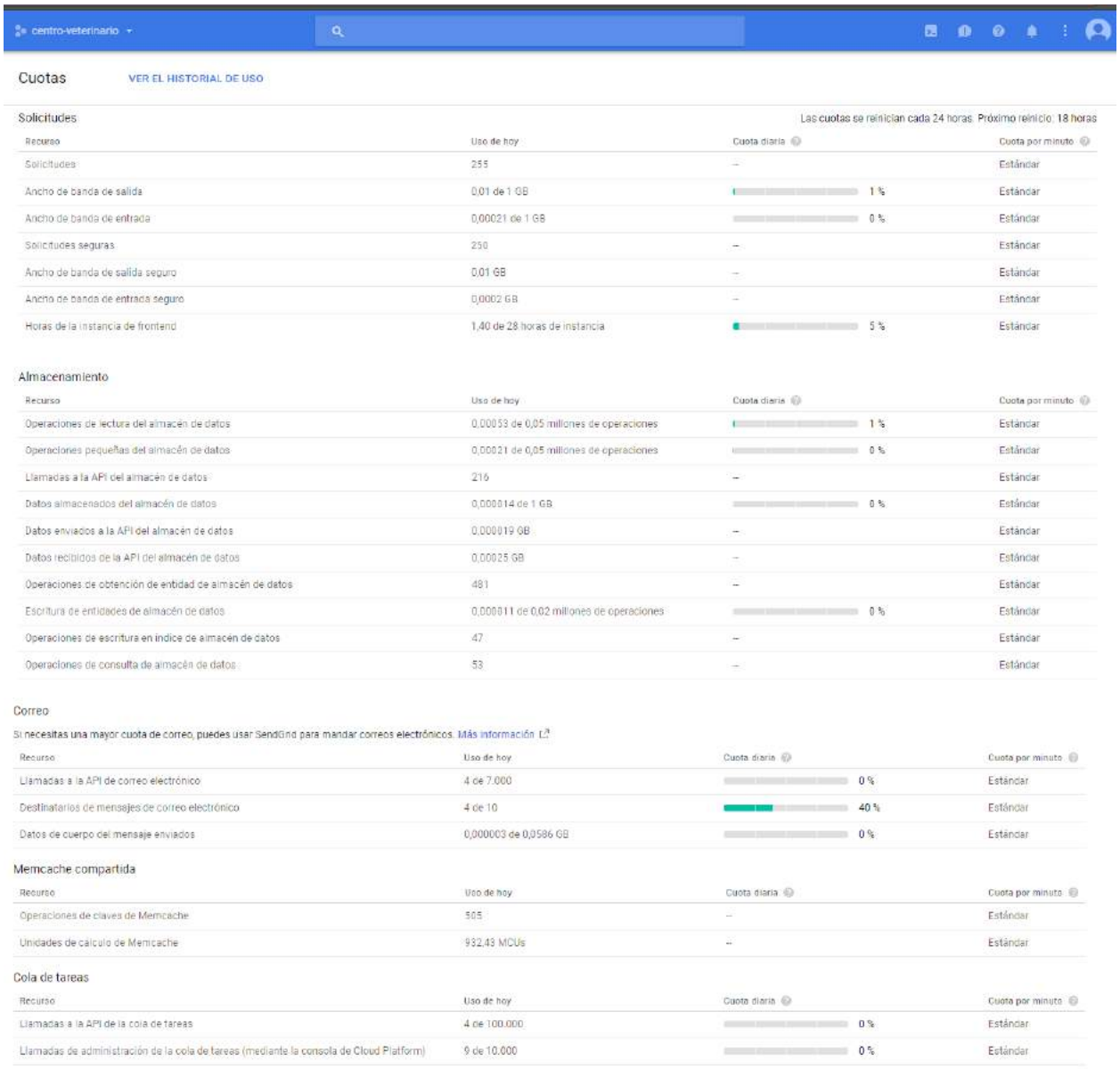

Figura 32. *Captura de las cuotas de App Engine.*

- Desde el **Panel de Control del Datastore**, podremos visualizar información relacionada con las entidades y los índices de nuestra aplicación:

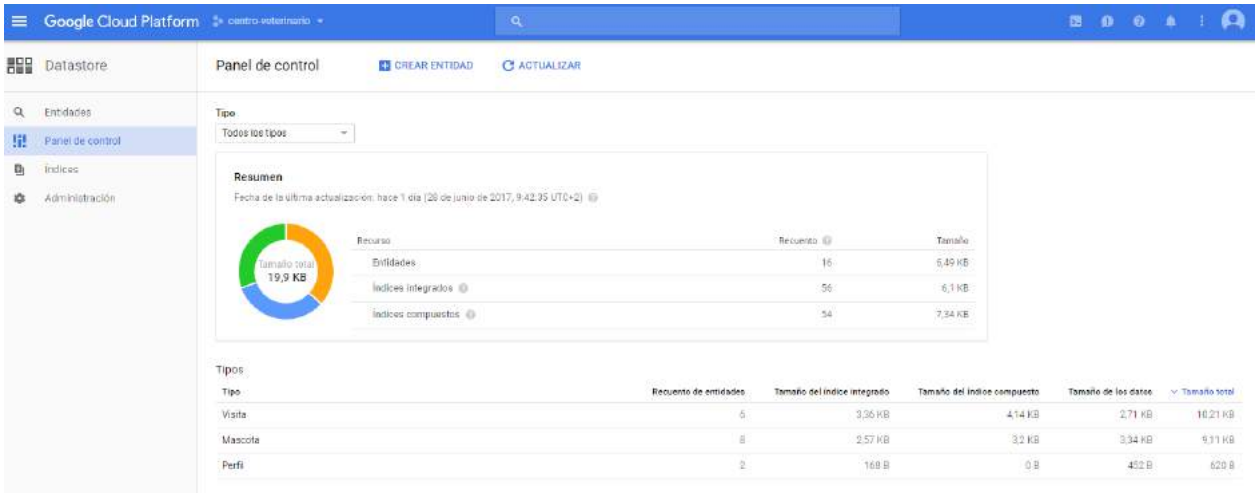

Figura 33. *Captura del panel de control que proporciona Datastore.*

- Desde la opción de **Logging de GCP**, podremos visualizar los registros que genera nuestra aplicación y comprobar que todo funciona correctamente o detectar problemas:

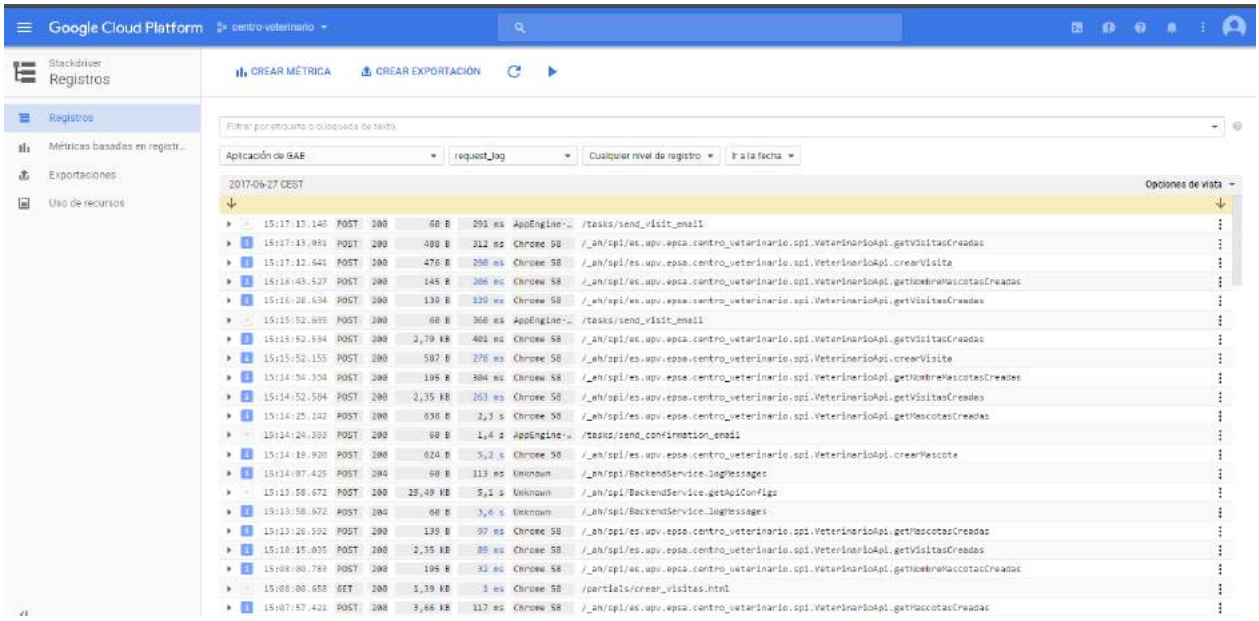

Figura 34. *Captura de los registros de la aplicación.*

6. Conclusiones

6.1 Conclusiones personales

Durante la realización de este proyecto de final de grado he podido comprobar la **importancia** que está alcanzando y que alcanzará la **computación en la nube** y la cantidad de ventajas que supone el utilizar las infraestructuras de grandes compañías como Google para las empresas.

Además, me ha aportado un sinfín de **satisfacciones personales,** aunque también hay que decir que han habido momentos muy complicados para poder conseguir el funcionamiento deseado.

A continuación, enumero todo lo que he **aprendido** a lo largo de este proyecto:

- La gran cantidad de servicios en la nube para todo tipo de soluciones informáticas y en especial para el desarrollo de aplicaciones web, así como a sintetizar la información más importante que nos proporcionan los diversos proveedores.
- A plantear y organizar el desarrollo completo de una aplicación web real.
- A llevar a cabo la gestión y desarrollo de un proyecto completo: desde la fase inicial de análisis de requisitos, pasando por el diseño de la aplicación completa, que ha conllevado desde la elección de la tecnología hasta el diseño final, y finalmente la implementación y testeo de la aplicación web.
- A tener en cuenta que la planificación de horas de programación debe ser con un amplio margen ya que es difícil que la implementación salga sin errores a la primera.
- A tomar decisiones para poder solucionar los problemas que van surgiendo a la hora de implementar la aplicación y buscar soluciones alternativas para conseguir el objetivo inicial.
- A utilizar diversas fuentes y la formación que ofrece Google dirigida a la creación de aplicaciones web para App Engine, así como a descubrir el potencial de Google Cloud Platform.
- Además, he ampliado mis conocimientos, adquiridos a lo largo del grado, en cuanto a la programación en Java, al uso de HTML y CSS y he aprendido a utilizar Bootstrap y AngularJS.

6.2 Futuras líneas de desarrollo

Teniendo en cuenta que la aplicación está en estos momentos está registrada en una cuenta gratuita de la plataforma de Google Cloud el siguiente paso sería **registrar una cuenta de facturación** para poder pasar a disfrutar al 100% de todos los servicios que nos proporciona Google Cloud Platform.

Aunque actualmente la aplicación posee un diseño responsive y eso hace que desde cualquier dispositivo con un navegador se pueda utilizar, en un futuro se podría diseñar su correspondiente **aplicación móvil** para Android e iOS.

El resto de **futuras implementaciones** de mejora serían **a petición del cliente**, por ejemplo, la posibilidad de que las tablas con los listados de mascotas y visitas fueran ordenables desde las cabeceras de las misma o el incluir más filtros para la búsqueda de registros.

Por otra parte, se podría llevar a cabo la **ampliación de la funcionalidad de la aplicación**, para que desde cualquier centro veterinario se pudiera acceder a todas las mascotas y visitas, aunque los usuarios únicamente pudieran visualizar las suyas.

Ampliar las entidades de las visitas para que los veterinarios pudieran adjuntar diversa información, por ejemplo, los pdfs con los resultados de análisis, radiografías, etc…

Finalmente, la **posibilidad de coordinar el calendario del cliente y del veterinario** con la opción de compartir un calendario de tal forma que el veterinario pudiera ver todas las próximas visitas del centro veterinario en cuestión mientras que los clientes únicamente vieran sus propias visitas.

7. Bibliografía

1. Luis Joyanes Aguilar, *"Computación en la nube: Estrategias de cloud computing en las empresas"*, 2013 Marcombo S.A., ISBN: 978-84-267-1893-8
2. Google, "*Cloud Computing, servicios de alojamiento y APIs de Google Cloud | Google Cloud Platform*", online
3. Google, "*Products & Services*", online
4. Google, "*Cloud Compute Products*", online
5. Google, "*Cloud Storage Products*", online
6. Google, "*Cloud Networking Products*", online
7. Google, "*Cloud Big Data Products*", online
8. Google, "*Cloud Machine Learning Products*", online
9. Google, "*Cloud Management Tools*", online
10. Google, "*Cloud Developers Tools*", online
11. Google, "*Cloud Cloud Platform Security*", online
12. Google, *"App Engine"*, online
13. Google, "*The App Engine Standard Environment*", online
14. Google, "*Google App Engine Documentation*", online
15. Google, "*App Engine Flexible Environment*", online
16. Google, *"Choosing an App Engine Environment"*, online
17. Google, "*App Engine Service Level Agreement (SLA)*", online
18. Google, "*App Engine Locations*", online
19. Amazon, "*AWS | Cloud Computing - Servicios de informática en la nube*", online
20. Amazon, "*Productos y servicios de la nube: Amazon Web Services (AWS)*", online
21. Amazon, *"Cloud Compute with AWS"*, online
22. Amazon, *"Almacenamiento en la nube con AWS"*, online
23. Amazon, *"Bases de datos en la nube con AWS"*, online
24. Amazon, *"Productos de redes con AWS"*, online
25. Amazon, *"Herramientas para desarrolladores con AWS"*, online
26. Amazon, *"Herramientas de administración de AWS"*, online
27. Amazon, *"Seguridad, identidad y conformidad en la nube con AWS"*, online
28. Amazon, *"Servicios de análisis con AWS"*, online
29. Amazon, *"Amazon AI"*, online
30. Amazon, *"Servicios para móviles de AWS"*, online
31. Amazon, *"Internet de las cosas"*, online
32. Amazon, *"Amazon Lumberyard"*, online
33. Amazon, "*AWS Elastic Beanstalk*", online
34. Amazon, *"Detalles del producto AWS Elastic Beanstalk"*, online
35. Amazon, "*AWS Elastic Beanstalk - Developer Guide (API Version 2010-12-01)*", online

36. Microsoft, "*Microsoft Azure: plataforma y servicios de informática en la nube*", online
37. Microsoft, "Productos de Azure", online
38. Microsoft, "*¿Qué es Servicios de aplicaciones de Azure?*", online
39. Lauren Cooke, "*5 Open Source Cloud Platforms for the Enterprise*", 24 de Mayo 2016, online
40. Red Hat, "*Red Hat OpenShift*", online
41. Cloud Foundry Foundation, "*Cloud Foundry*", online
42. Globo Dev, "Tsuru", online
43. GigaSpaces Technologies, "*Cloudify*", online
44. Red Hat, "*Red Hat OpenShift Online*", online
45. Red Hat, "*Red Hat OpenShift Container Platform*", online
46. Google, "*Architecture: Web Application on Google App Engine*", online
47. Google, "*App Engine Pricing*", online
48. Google, "*Quotas*", online
49. Google, "*Google Cloud Platform - Versión gratuita*", online
50. Google, "*Google Cloud Platform Pricing Calculator*", online
51. Google, "*Google APIs Explorer*", online
52. Google, "*Google Cloud Platform Documentation*", online
53. Google, "*Google App Engine Documentation*", online
54. Coursera, "*Coursera*", online
55. Udacity, "*Udacity*", online
56. Coursera, "*Welcome to GCP Fundamentals*", online
57. Udacity, "*Developing Scalable Apps in Java with Google App Engine*", online
58. Google, "*Using Custom Domains and SSL*", online
59. Google, "*Google Plugin for Eclipse*", online
60. Amazon, "*Architectural Overview*", online
61. Amazon, "*Your Elastic Beanstalk Environment's Domain Name*", online
62. Amazon, "*AWS Toolkit for Eclipse*", online
63. Amazon, "*Capa gratuita de AWS*", online
64. Amazon, "*Precios de AWS Elastic Beanstalk*", online
65. Amazon, "*Precios de Amazon EC2*", online
66. Amazon, "*Precios de Amazon S3*", online
67. Amazon, "*Precios de Amazon DynamoDB*", online
68. Amazon, "*Amazon Web Services Simple Monthly Calculator*", online
69. Amazon, "*AWS Total Cost of Ownership (TCO) Calculator*", online
70. Sara Ashley O'Brien, "*Here's why the internet didn't work very well today*", 28 de febrero de 2017, online

71. Raúl Álvarez, "*No, no es que tu conexión vaya mal, es que medio Internet está fallando tras una caída en Amazon S3"*, 28 de febrero de 2017, online
72. Expansion.com, "*La nube de Amazon se cae y provoca problemas en las principales webs de EEUU"*, 28 de febrero de 2017, online
73. Amazon, "*Summary of the Amazon S3 Service Disruption in the Northern Virginia (US-EAST-1) Region*", online
74. Microsoft, "*Improve scalability in a web application"*, online
75. Microsoft, *"Cree su cuenta gratuita de Azure hoy mismo"*, online
76. Microsoft, *"App Service Precios"*, online
77. Microsoft, *"Calculadora de precios"*, online
78. Microsoft, *"Documentación de App Service"*, online
79. Microsoft, *"Asignación de un nombre de dominio personalizado a una aplicación de Azure "*, online
80. Microsoft, *"Azure Toolkit for Eclipse"*, online
81. PayScale, *"Salary Survey, Salaries, Wages, Compensation Information and Analysis"*, online
82. Kayle Roche y Jeff Douglas,"*Beginning Java™ Google App Engine"*, 2009 Apress, ISBN-13 (electronic): 978-1-4302-2554-6
83. Dan Sanderson, *"Programming Google App Engine with Java"*, 2015 O'Reilly, ISBN: 1-4919-0345-7, 1-4919-0020-2
84. Udacity, Curso: "*Developing Scalable Apps in Java*", Lecciones 3 y 4, online
85. Google, "*Cloud Datastore*", online
86. Google, *"Cloud BigTable"*, online
87. Google Developers, *"Google I/O 2009 - Java Persistence & App Engine Datastore",* online
88. Google, "*GQL Reference*" ,online
89. Google, *"Versión de prueba y gratuita de GCP",* online

8. Acrónimos

- ***ACID:*** (*Atomicity, Consistency, Isolation, Durability).* Atomicidad, Consistencia, Aislamiento y Durabilidad en las bases de datos.
- ***API****: (Application Programming Interface)* Interfaz de Programación de Aplicaciones.
- ***AWS:*** *Amazon Web Services.*
- ***BACK-END:*** capa de acceso a datos de una aplicación web.
- ***CLI:*** interfaz de la línea de comandos.
- ***CSS:*** (*Cascading Style Sheets).* Lenguaje de hojas de estilos para ajustar el diseño y apariencia de las hojas HTML y XHTML.
- ***DNS****: (Domain Name System)* Sistema de nombres de dominio.
- ***EB:*** CLI de alto nivel diseñado específicamente para Elastic Beanstalk.
- ***EIS:*** *(Enterprise Information System).* Sistema de Información Empresarial.
- ***FRONT-END:*** capa de presentación (interfaz) de una aplicación web.
- ***GCP:*** *Google Cloud Platform.*
- ***GQL:*** lenguaje de consulta SQL ajustado por parte de Google para poder utilizarlo en sus bases de datos NoSQL.
- ***HTML:*** *(HyperText Markup Language).* Lenguaje de marcado que se utiliza para el desarrollo de páginas de web.
- ***iOS:*** Sistema operativo móvil de Apple.
- **JDK:** *(Java Development Kit)*. Software que proporciona herramientas de desarrollo para crear programas con Java.
- ***JDO:*** *(Java Data Objects).* API que proporciona una forma estándar para la persistencia de objetos en Java.
- ***JPA:*** *(Java Persistence API).* API para poder relacionar bases de datos y sistemas orientados a objetos.
- ***NoSQL:*** bases de datos distribuidas no relacionales.
- ***SDK:*** (*Software development kit)* Kit de desarrollo de software.
- ***SMB:*** *(Server Message Block).* Protocolo de red para compartir archivos e impresoras.
- ***SQL:*** *(Structured Query Language). Lenguaje de consulta para bases de datos relacionales.*
- ***URL:*** *(Uniform Resource Locator).* Localizador Uniforme de Recursos.
- ***XML:*** *(Standard Generalized Markup Language).* Lenguaje que permite la organización y etiquetado de documentos.

9. Anexo

9.1 Códigos

Todo el código de la aplicación web desarrollada se anexa en la carpeta comprimida (COGIGO_ANA_OLCINA_VALERO.zip) junto a esta memoria.